SNS 109020200-TD0001-R00

SNS Timing System Technical Description

April 2003

SPALLATION NEUTRON SOURCE
A U.S. Department of Energy Multilaboratory Project
Argonne National Laboratory • Brookhaven National Laboratory • Thomas Jefferson National Accelerator Facility • Lawrence Berkeley National Laboratory • Los Alamos National Laboratory • Oak Ridge National Laboratory
SNS

This report was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or any agency thereof.

Eric Bjorklund, bjorklund@lanl.gov
Dave Thompson, dht@ornl.gov

Date Published: April 2003

Prepared for the
U.S. Department of Energy
Office of Science

UT-BATTELLE, LLC
managing
Spallation Neutron Source activities at
Argonne National Laboratory
Brookhaven National Laboratory
Thomas Jefferson National Accelerator Facility
Lawrence Berkeley National Laboratory
Los Alamos National Laboratory
Oak Ridge National Laboratory
under contract DE-AC05-00OR22725
for the
U.S. DEPARTMENT OF ENERGY

Table Of Contents

1 SYSTEM OVERVIEW

The SNS timing system consists of two data transmission links distributed to all accelerator systems. The *Event Link* transmits 8-bit timing events that define the SNS *Machine Cycle*. The events on the event link are synchronized with the accumulator ring RF frequency. The *Real Time Data Link* (RTDL) transmits a series of 24-bit data frames prior to the beginning of a machine cycle. These data frames contain information about the next cycle such as the time of day, the type of pulse to be delivered (pulse *flavor*), information on whether the preceding pulse was good etc. The SNS *Timing Gate Trigger* module (V124S) translates timing events into TTL gates. Each gate can have individually programmed parameters such as width, delay (from the triggering event), and polarity. The SNS *Utility Module* listens to both the event link and the RTDL. The utility module can interrupt the IOC on the occurrence of specified events. It can also provide the IOC with data from the RTDL.

The figure below shows a block diagram of the SNS timing system, its various components, and their relationships. The exact function of each of these components is described in greater detail in the rest of this document.

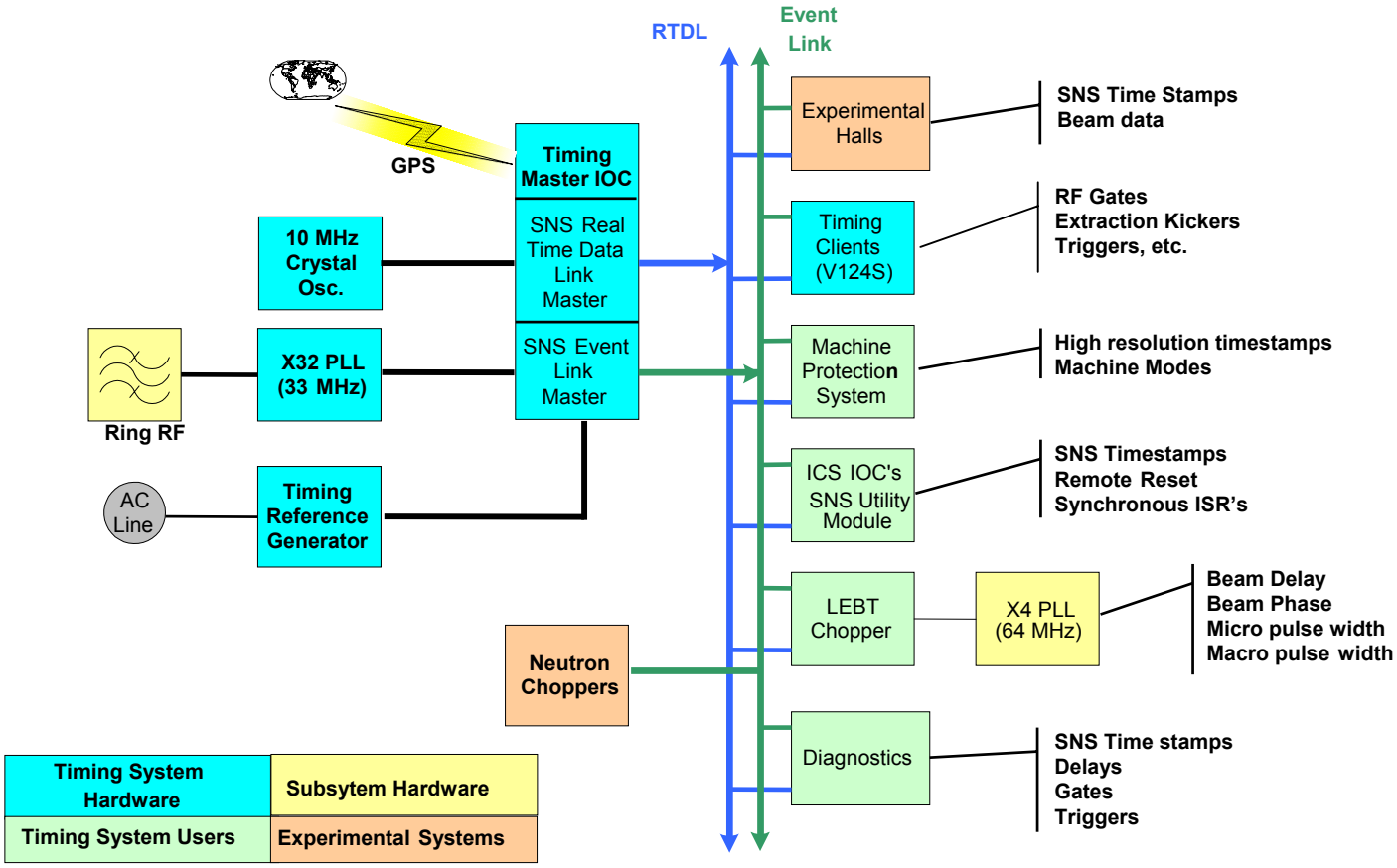


Figure 1
Timing System Components

1.1 TIMING SYSTEM CONCEPTS

1.1.1 Machine Cycle Time Line

The fundamental repetition rate of the SNS timing system is 60 Hertz. The starting point for a machine cycle is a 60 Hz "smoothed" (or "damped") signal derived from the positive zero crossing of the AC line frequency. The maximum repetition rate for delivering beam to the first neutron production target is 60 Hertz. The maximum repetition rate for delivering beam to the second neutron production target is 20 Hertz. When the second target is implemented, there is some consideration that the 20 Hz of beam to target 2 might be delivered from the negative zero-crossing, allowing us to continue to deliver 60 Hz of beam to target 1. Consequently, the timing system design must allow for the possibility of 120 Hz operations, which means that the maximum length of a machine cycle can only be 8.3 milliseconds.

The SNS machine cycle is subdivided into two sections, a "Time Critical" section and a "Non-Critical" section. The time critical section begins at the start of the cycle, includes the acceleration of beam in the linac, the accumulation of beam in the ring, and ends with its extraction from the ring and delivery to the neutron production target. The parts of a machine cycle are defined by *Events,* which are broadcast on the SNS *Event Link.* Events can be either hardware or software generated. Software-generated events are not allowed to occur during the time-critical portion of the machine cycle, since they might interfere with the delivery of time-critical hardware events.

The most time-critical event in the machine cycle is the "Extract" event. This is the event synchronizes the *Extraction Kicker* magnets and the *Neutron Choppers*. The neutron choppers are located between the neutron production target and the experimental instruments. They come in two basic flavors, *T-Zero* and *E-Zero*. The job of the T-Zero choppers is to "chop" out the initial high-energy garbage that gets ejected immediately after the protons hit the neutron production target. These choppers are high-inertia and cannot respond quickly to changes in their synchronization pulses. One of the jobs of the timing system, therefore, is to try to keep the Extract event as stable in time as possible, in spite of fluctuations in the AC line frequency. Another job of the timing system is to try to keep the machine cycle from getting too far out of phase with the AC line frequency. We discuss the management of these two conflicting goals in subsequent sections of this document.

It is not desirable to leave beam in the accumulator ring any longer than necessary. Therefore (and unlike more traditional accelerators), the injection of beam into the accumulator ring is timed such that it always ends slightly before the "Extraction" event. A special "End-Inject" event occurs just before Extract and defines the fixed point where the accelerator RF and beam gates end. If more beam is desired, the beam and RF gates get longer by growing toward Cycle-Start. If less beam is desired, the beam and RF gates compact toward the End-Inject event.

The picture below illustrates the basic layout of the SNS machine cycle.

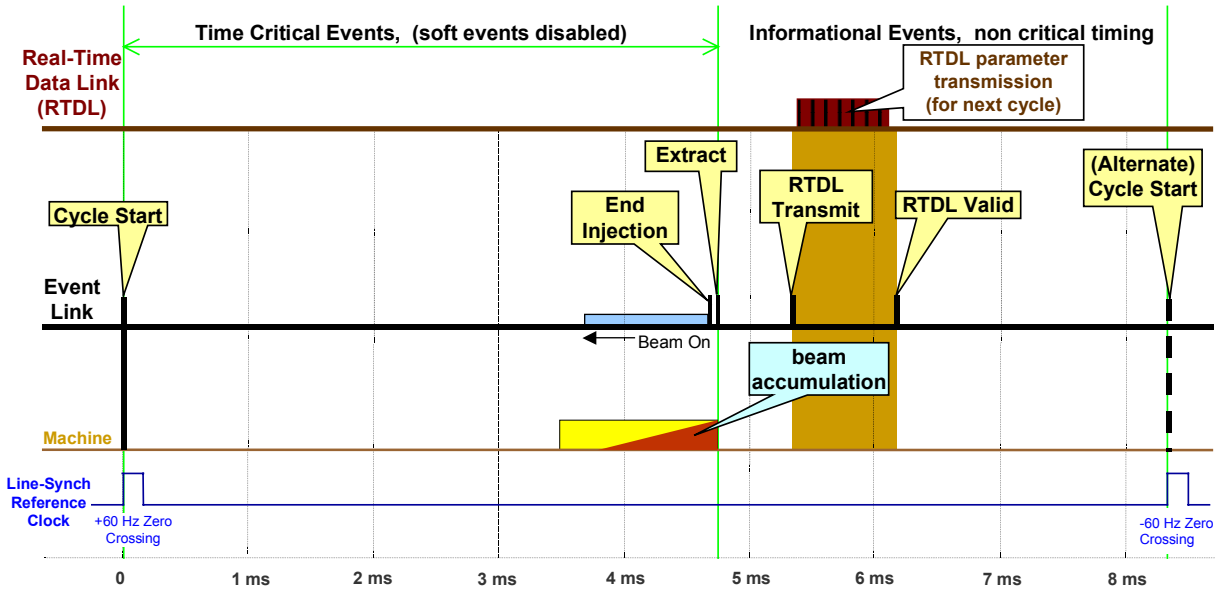


Figure 2
Basic Machine Cycle Time Line

The time-critical portion of the cycle begins with the "Cycle-Start" event and ends with the "Extract" event. Software-generated events are not allowed to occur within this period of time. If software does request an event during the time-critical portion of the machine cycle, it will be queued in a FIFO and not released until after the Extract event. Events that control the firing of the accelerator RF gates, front-end choppers, and data acquisition diagnostic gates also occur within the time-critical section.

The non-time-critical portion of the machine cycle begins after the Extract event and continues until the next Cycle-Start (or Alternate Cycle-Start, if running in 120 Hz mode). During this time, software-generated events are allowed to occur. Software-generated events announce non-time-critical activities such as global error-counter clearing, or the start or stop of a new rep-rate pattern generation. This is also the period when the *Real Time Data Link* (RTDL) frames for the next machine cycle are transmitted. Shortly after the Extract event, the timing master sequencer program computes the appropriate RTDL frame values for the next machine cycle and then generates the "RTDL Xmit" event. The RTDL-Xmit event triggers a gate that instructs the RTDL encoder module to transmit the current set of frames over the link. After the entire set of RTDL frames has been sent, the "RTDL Valid" event signals that all frames have been sent and are available for use by client IOCs or diagnostic NADs.

1.1.2 Turns and Sub-Revolutions

The fundamental unit of the SNS timing system is the *Turn*, which is defined as the amount of time it takes the beam to travel around the accumulator ring one time. For rough approximation purposes, one

turn is approximately equivalent to one microsecond. In reality the value of a turn can vary between 911 and 974 microseconds depending on the energy of the beam and the size of the orbit. The dominant term is the beam energy which can vary between 842 MeV and 1.3 GeV. As shown above in Figure 1, the timing system clock is derived from the RF signal that maintains the extraction gap in the beam. The ring RF frequency is multiplied by 32, which gives the timing system clock a granularity of about 29.5 nanoseconds at 1GeV. Each timing system clock tick is referred to as a *Sub-Revolution*.

Using the ring revolution frequency as the timing system clock allows the neutron choppers to be precisely synchronized with the extraction kick. It also allows us to tune the ring, set different energies, and adjust the size of the orbit without having to constantly change all the timing parameters.
The disadvantage, of course, is that if you have any "wall clock" timing constraints, you now have to deal with "error bars" around your settings. For example, if you want to guarantee that the High-Power RF gates turn on at least 100 microseconds before the Low-Level RF gates, you need to set the difference to 110 turns in order to accommodate the worst-case (fastest) clock speed at 1.3 GeV.

The table below gives the relationship between beam energy, the ring RF frequency, and the turn and sub-revolution times:

Beam Energy	Mini-pulse Length (ns)	Gap Length (ns)	Revolution Period (ns)	Revolution Frequency (MHz)	Clock Frequency (MHz)	Beta	Sub-Revolution Period (ns)
842 MeV	655.1	324	973.4	1.027,323	32.874,340	84.984 %	30.42
1.0 GeV	630.3	315	945.4	1.057,767	33.848,545	87.503 %	29.54
1.3 GeV	607.4	304	911.2	1.097,502	35.120,070	90.790 %	28.47

Table 1
Beam Energy and Revolution Frequency

1.1.3 Macro Pulses, Mini Pulses, and Micro Pulses

An SNS beam pulse has three levels of structure. The highest level is the *Macro Pulse,* which is defined by timing system gates. A macro pulse can be anywhere from 1 to 1060 turns (about 1 millisecond) long. The size of the macro pulse determines how long the accelerator RF needs to be on. It also determines the intensity of the beam delivered to the neutron production target.

When the beam enters the accumulation ring, it is compressed from the approximately one millisecond macro pulse into an approximately on microsecond *Mini Pulse*. In order for this to work, the macro pulse has to have a structure imposed on it in which there is a "gap" of about 300 nanoseconds at the start of each turn. This is known as the *Extraction Gap*. When it comes time to extract the beam from the ring, the extraction kickers turn on during the 300 nanosecond extraction gap. This allows them to be at full power by the time beam arrives and prevents "beam spraying". The mini pulse structure is imposed by the LEBT chopper, which gets its clock from the timing system.

The finest level of structure is the *Micro Pulse,* which reflects the accelerating structure imposed by the linac RF. The LEBT chopper can "ramp up" the beam intensity from the start to the end of the macro pulse by selectively "chopping out" selected mini pulses.

The figure below shows the relationship between the macro pulse, mini pulse and micro pulse structures.

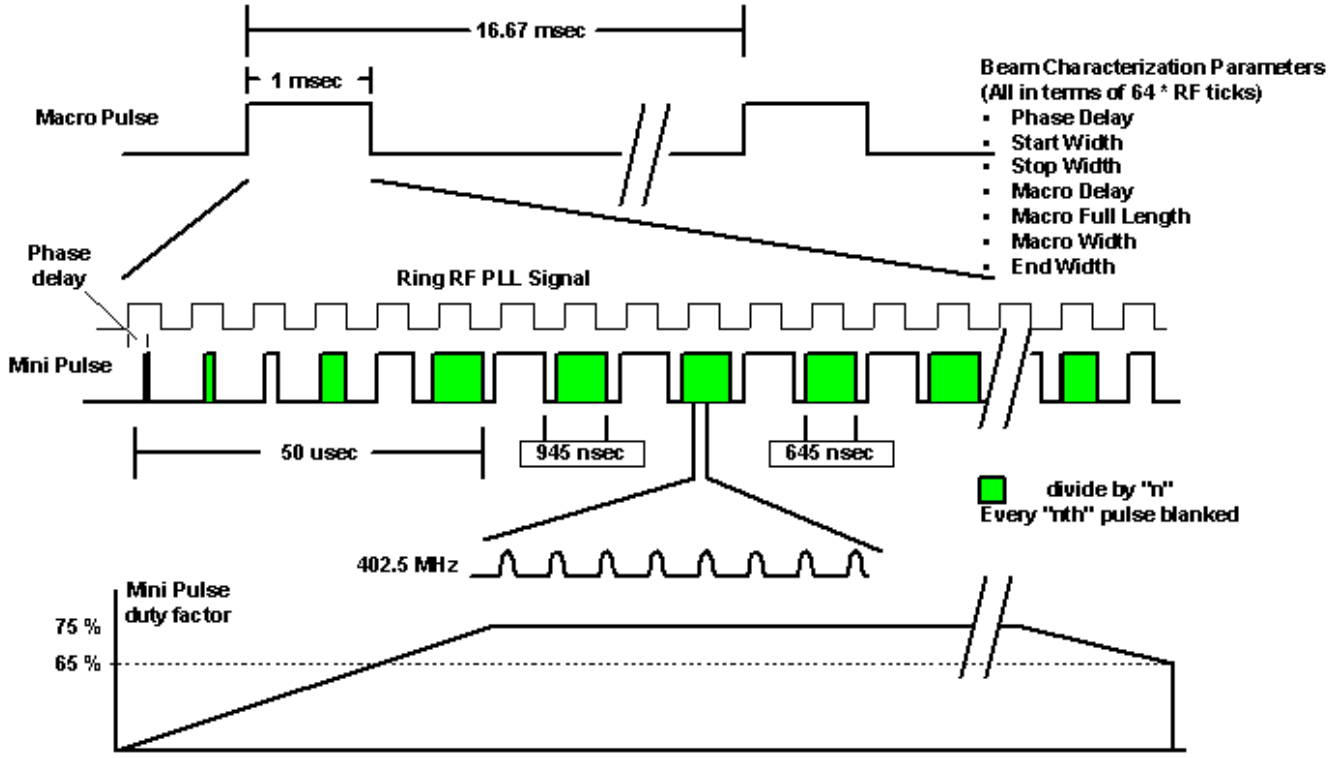


Figure 3
Macro, Mini, and Micro Pulses

1.1.4 Pulse Flavors

Each machine cycle has a *Pulse Flavor* associated with it. The pulse flavor for the next machine cycle is transmitted on the RTDL There are eight possible flavors for a machine cycle. These are:

0 No Beam
1 Normal Beam (Target 1)
2 Reserved for Target 2
3 10 μSecond Diagnostic Pulse
4 50 μSecond Diagnostic Pulse
5 100 μSecond Diagnostic Pulse
6 Special Physics Pulse 1
7 Special Physics Pulse 2

Beam flavors are used by the LEBT chopper to determine what the chopping pattern should be for the next cycle. They could also be used by various systems, such as Low-Level RF, to do adaptive tuning based on what kind of beam is being delivered. Using the SNS utility module, an IOC application can set up records to process only when beam of the particular flavor they are interested in has been scheduled.

The first 6 flavors are called "Normal" flavors. They reflect the current type of beam being produced and are usually determined from the MPS *Beam Mode*. The last two flavors (6 and 7) are called "Cycle Stealing" flavors. If they occur at all, they only occur at low repetition rates – typically 0.1 Hertz. They "steal" cycles from the normal flavored pulses. The stolen cycles have special parameters set by the LEBT chopper and are used for on-line accelerator physics experiments.

"No Beam" (flavor 0) means that no beam is scheduled for the next cycle. This could be because beam has been turned off, or the MPS beam mode is either "Off", "Standby" or "MPS Test", or the MPS

is not made up for the current Beam/Machine mode. It could also mean that the beam is running at a repetition rate under 60 Hz and the next cycle is not one scheduled for beam.

"Normal Target 1" (flavor 1) means that the full intensity beam is scheduled for the next cycle. In order for the flavor to be "1", the MPS beam mode needs to be either "1 Millisecond" or "Full Power". This does not mean that the actual beam gate needs to be 1 millisecond long.

Flavor 2 is reserved for the second target. It has the same properties as flavor 1, but reflects the parameters of the Target 2 beam. If the timing system is changed to operate at 120 Hz, then flavor 2 cycles are only scheduled on "alternate" (negative zero-crossing) cycles. Flavor 2 becomes the "other" normal flavor in this case. If the timing system continues to run at 60 Hz, then flavor 2 steals cycles from flavor 1 (up to 20 Hz).

Flavors 3, 4, and 5 correspond to the MPS beam modes for 50 μSecond, 100 μSecond, and 100 μSecond, diagnostic modes respectively. These modes reflect restricted intensity conditions due to the use of intercepting diagnostics. Consequently, these flavors can not be "shared" with each other or the other "normal" flavors. Neither can they steal pulses from the other normal flavors.

Flavors 6 and 7 are for intended for on-line accelerator physics experiments. They are intended to run at low rep-rates (0.1 Hz) so that they do not noticeably interfere with normal beam. They may steal cycles from any of the other "normal" flavors, provided their pulse width does not violate the current MPS operating mode.

1.2 EPICS SOFTWARE

The EPICS software for the SNS timing system is divided into two categories. The *Timing Master* software is found under the $IOCTOP directory in $IOCTOP/timingMaster/production. This directory contains drivers, databases, device support, and operator interface screens for the hardware and software modules that are only found on the timing master IOC. The *Timing Client* software is found under the $SHARE directory in $SHARE/timing/production. This directory mostly contains the driver, device support, and database templates for the timing gate trigger module (V124S). The V124S module is the client IOC's primary interface to the timing system event link.

1.2.1 Client Software

The timing system client software mostly resides in the $SHARE/timing/production directory. The exception is the header file:

snsTiming.h

which resides in the utility module directory, $SHARE/utility/production/include. This file contains the definitions of the SNS event numbers and RTDL frames, along with other basic timing system parameters.

There is a bilateral dependency between the timing system software and the utility module software. The utility module software needs to know the SNS event number and RTDL frame definitions so that it will know when new RTDL information arrives and which frame numbers to extract the timestamp and beam flavor information from. The timing system, on the other hand, needs to use the utility module for various functions such as obtaining the ring revolution frequency from the RTDL so that it can convert from timing system units (turns and sub-revolutions) to microseconds. In an effort to avoid circular dependencies, the "snsTiming.h" file is located in the utility module directory.

When rebuilding the timing system client software (e.g. when upgrading to a new EPICS version), it is important to first rebuild the utility module directory and the Gensub directory.

The file, "snsTimingLib", in the "$SHARE/timing/production/bin/<*arch*>" directory contains the software needed for the timing gate trigger (V124S) module. IOC's that contain one or more V124S modules should load both "snsTimingLib" and "snsUtilLib" (from "$SHARE/utility/production/bin/<*arch*>").

1.2.2 Timing Master Software

The timing master software is found under the "$IOCTOP/timingMaster/production" directory. This directory contains several application sub-directories, not all of which need to be present for any given installation. The application directories are:

- **timingLib** – This directory contains drivers, device support, database templates, and OPI screens common to all the other application directories. This application should be present for all installations.
- **timingMaster** – This application contains the timing master application software, databases and OPI screens for the timing master IOC that runs at the SNS site.
- **devApp** – This application contains the timing master application software, databases and OPI screens for the development timing master IOC that runs in the SNS controls lab at 701 Scarboro Road.
- **dev2App** – This application contains the timing master application software, databases and OPI screens for the hot spare timing master IOC that runs at the SNS site.
- **jlabApp** – This application contains the timing master application software, databases and OPI screens for the timing master IOC that runs at the Jefferson Laboratory test stand.
- **lanlApp** – This application contains the timing master application software, databases and OPI screens for the timing master IOC that runs at Los Alamos.

2 TIMING MASTER IOC

This section describes the timing master IOC as it is implemented at the SNS site. Some configuration differences may exist between this and the other development and test timing masters.

The event link and RTDL are produced by the SNS Timing Master IOC. The Timing Master IOC consists of two 21-slot VME64X crates bridged together with an SBS Technologies Model 418 VMEbus repeater. The top crate contains the modules for generating the RTDL. The lower crate contains the processor and the modules for generating the event link. The lower crate also contains most of the other timing system hardware interface modules. The figure below illustrates the layout of a fully loaded set of timing master IOC crates.

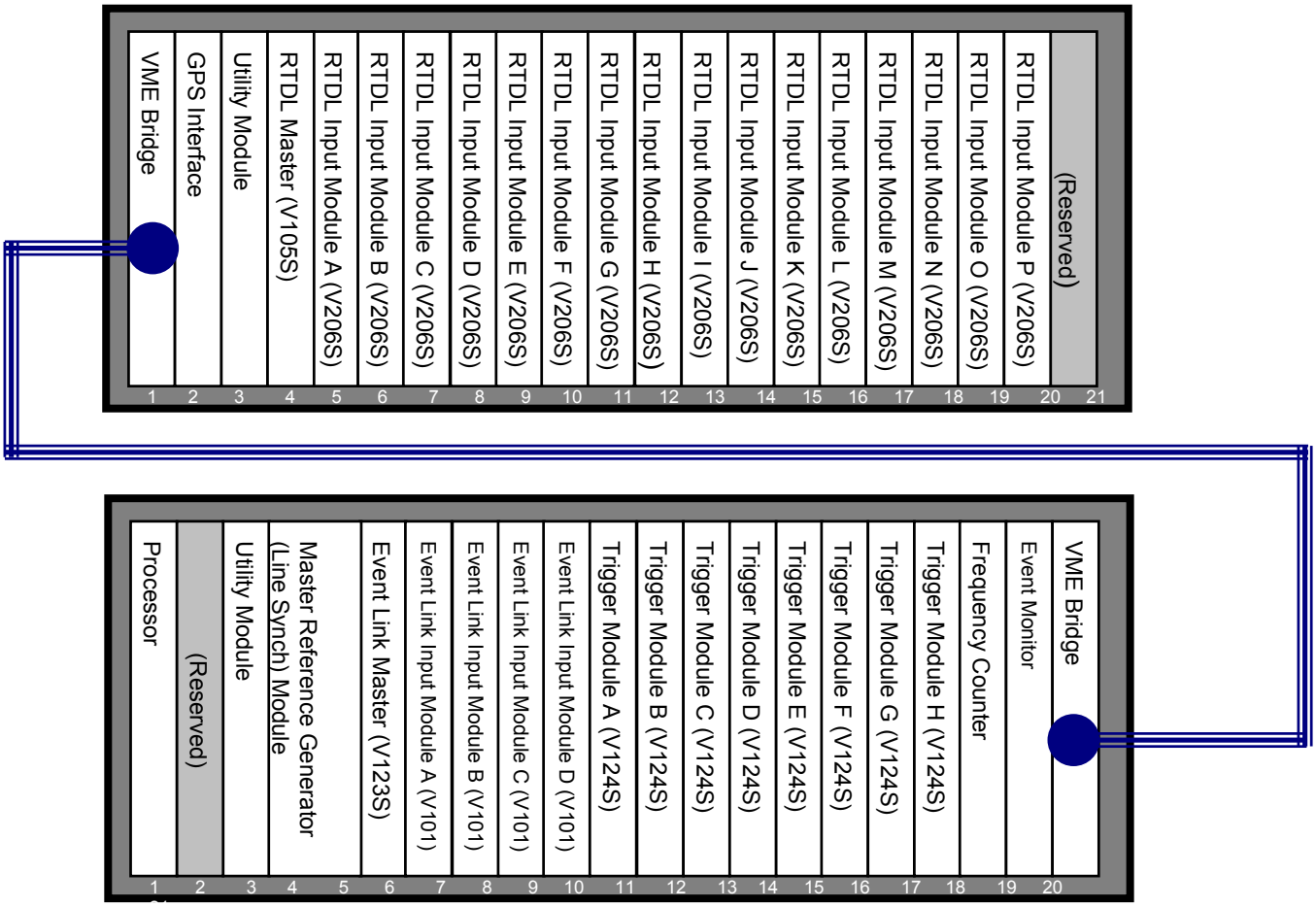


Figure 4
Timing Master Crate Layout

The timing master IOC contains the following components:

- **Timing Gate Triggers:** Up to 8 V124S timing gate trigger modules provide up to 64 TTL timing gates. These gates are mostly used to generate "*Hardware Timing Events*" – i.e. those events with strict timing requirements that define the SNS machine cycle sequence. Some of these timing gates are also used to trigger other components. For example, one gate triggers the RTDL transmission for the next cycle, another gate freezes the current GPS time so that it can be sent out on the RTDL, etc.

- **Event Link Encoder:** Generates the SNS event link that is distributed throughout the accelerator. The carrier signal frequency is 16 X the ring revolution frequency (approximately 17 Mhz).
- **Master Reference Generator:** Monitors the AC line frequency and generates a "stable" 60 Hz reference pulse designed to not vary any faster than the neutron choppers can stay in synch with, while at the same time not varying to far from the actual zero-crossing so that the klystrons can stay relatively line-locked with the AC. The intersection of the Master Reference Generator's reference pulse with the 32 X Frev event link clock defines the start of a "*Ring Revolution*".
- **Ring RF Counter:** Measures the frequency of the 32 X Frev timing system clock. This value is broadcast on the RTDL and used to convert timing system units (turns and subrevolutions) into microseconds.
- **RTDL Encoder:** Generates the Real-Time Data Link that is distributed throughout the accelerator. The carrier signal frequency is 10 Mhz and is independent of the ring revolution frequency.
- **GPS Interface:** System time is acquired from a GPS system and broadcast on the RTDL each cycle. The GPS time is "frozen" at each Cycle-Start event, incremented by 16.667 milliseconds, and broadcast on the RTDL as the time of the next Cycle-Start event.
- **Event Monitor:** Timing system diagnostic. Monitors and timestamps events on the event link. Timing master can compare these events with what it thought it sent last cycle.
- **Utility Module:** Monitors the health of the VME crate. Also provides an interface from the timing master IOC to the Machine Protection System (MPS).

2.1 HARDWARE LAYOUT

The following VME address and vector ranges are allocated within the timing master IOC:

A16:	100-1FF	GPS System	Source for RTDL timestamp frames
	800-FFF	Joerger Counter	Measures the ring RF revolution period.
	1000-1FFF	Event Link System	Space reserved for Event Link Master and up to 8 input modules (only 4 input modules are possible in current system)
	2000-27FF	Event Link Monitor	Tentative allocation. Module not implemented yet.
	2800-2FFF	Master Reference Generator	Tentative allocation. Module not implemented yet
	7000-AFFF	Timing Gate Generators	Space reserved for 8 V124S modules (64 gates)
	BE00-FFFF	RTDL System	Space reserved for RTDL Master and up to 32 RTDL input modules (only 16 input modules are possible without adding a third crate).
A24:	4000-A7FF	Utility Modules	Space reserved for 2 Utility Modules (1 per crate)
Vectors:	20-28	Event Link System	Vectors for Event Link Master and up to 8 input modules (only 4 input modules are currently possible)
	30-3E	Timing Gate Generators	Space reserved for 8 V124S modules.
	40-4A	Utility Modules	Space reserved for 2 Utility modules
	50-70	RTDL System	Space reserved for RTDL Master and up to 32 RTDL input modules (only 16 input modules are possible without adding a third crate)

Figure 5
Timing Master VME Address and Vector Assignments

The module, slot, and address assignments for the timing master IOC are listed below:

Crate 1 (Lower Crate)				
Slot	**Module**	**Address**	**Vector**	**Description**
1	Motorola MV2100	NA	NA	Timing Master IOC Processor
2	(Reserved)			(Good place for a bus analyzer)
3	Brookhaven Utility Module	A24 4000-67FF	40	Monitors VME crate environment. Monitors RTDL transmissions. Input from MPS.
4-5	LANL/Brookhaven Master Reference Generator	A16 (??) 2800-2FFF	NA	Monitors AC line frequency and provides stable "Cycle-Start" trigger. Not implemented yet.
6	Brookhaven V123S	A16 1000-17FF	24*	Event Link Master. Generates SNS Event Link carrier and broadcasts 12-bit timing events.
7	Brookhaven V101S	A16 1800-187F	20*	Event Link Input Module A. Generates SNS events from 16 TTL gate inputs.
8	Brookhaven V101S	A16 1880-18FF	21*	Event Link Input Module B. Generates SNS events from 16 TTL gate inputs.
9	Brookhaven V101S	A16 1900-197F	22*	Event Link Input Module C. Generates SNS events from 16 TTL gate inputs.
10	Brookhaven V101S	A16 1980-19FF	23*	Event Link Input Module D. Generates SNS events from 16 TTL gate inputs.
11	Brookhaven V124S	A16 7000-77FF	30	Trigger Module A. Generates TTL gates to trigger events and other system hardware
12	Brookhaven V124S	A16 7800-7FFF	32	Trigger Module B. Generates TTL gates to trigger events and other system hardware
13	Brookhaven V124S	A16 8000-87FF	34	Trigger Module C. Generates TTL gates to trigger events and other system hardware
14	Brookhaven V124S	A16 8800-8FFF	36	Trigger Module D. Generates TTL gates to trigger events and other system hardware
15	Brookhaven V124S	A16 9000-97FF	38	Trigger Module E. Generates TTL gates to trigger events and other system hardware
16	Brookhaven V124S	A16 9800-9FFF	3A	Trigger Module D. Generates TTL gates to trigger events and other system hardware
17	Brookhaven V124S	A16 A000-A7FF	3C	Trigger Module G. Generates TTL gates to trigger events and other system hardware
18	Brookhaven V124S	A16 A800-AFFF	3E	Trigger Module H. Generates TTL gates to trigger events and other system hardware
19	Joerger VS64	A16 800-FFF	NA	50 Mhz Counter Module. Measures the ring RF frequency from the timing system event link.
20	Brookhaven Event Monitor	A16 (??) 2000-27FF	(??)	Monitors and timestamps events on the event link. Timing system diagnostic. Not implemented yet.
21	SBS 418	NA	NA	VME Bridge Master
			* Vectors 20-24 are used by the event link system. Use vector 20 in the event link initialization call.	

Crate 2 (Upper Crate)				
Slot	**Module**	**Address**	**Vector**	**Description**
1	SBS 418	NA	NA	VME Bridge Slave
2	TrueTime VME-SG2	A16 100-1FF	NA	GPS Interface Module.
3	Brookhaven Utility Module	A24 8000-A7FF	NA	Monitors VME crate environment for Crate 2.
4	Brookhaven V105S	A16 BE00-BFFF	50	RTDL Master. Generates the SNS RTDL carrier and broadcasts 24-bit data frames every cycle.
5	Brookhaven V206	A16 C000-C1FF	51	RTDL Input Module A. Generates 8 RTDL frames.
6	Brookhaven V206	A16 C200-C3FF	52	RTDL Input Module B. Generates 8 RTDL frames.
7	Brookhaven V206	A16 C400-C5FF	53	RTDL Input Module C. Generates 8 RTDL frames.
8	Brookhaven V206	A16 C600-C7FF	54	RTDL Input Module D. Generates 8 RTDL frames.
9	Brookhaven V206	A16 C800-C9FF	55	RTDL Input Module E. Generates 8 RTDL frames.
10	Brookhaven V206	A16 CA00-CBFF	56	RTDL Input Module F. Generates 8 RTDL frames.
11	Brookhaven V206	A16 CC00-CDFF	57	RTDL Input Module G. Generates 8 RTDL frames.
12	Brookhaven V206	A16 CE00-CFFF	58	RTDL Input Module H. Generates 8 RTDL frames.
13	Brookhaven V206	A16 D000-D1FF	59	RTDL Input Module I. Generates 8 RTDL frames.
14	Brookhaven V206	A16 D200-D3FF	5A	RTDL Input Module J. Generates 8 RTDL frames.
15	Brookhaven V206	A16 D400-D5FF	5B	RTDL Input Module K. Generates 8 RTDL frames.
16	Brookhaven V206	A16 D600-D7FF	5C	RTDL Input Module L. Generates 8 RTDL frames.
17	Brookhaven V206	A16 D800-D9FF	5D	RTDL Input Module M. Generates 8 RTDL frames.
18	Brookhaven V206	A16 DA00-DBFF	5E	RTDL Input Module N. Generates 8 RTDL frames.
19	Brookhaven V206	A16 DC00-DDFF	5F	RTDL Input Module O. Generates 8 RTDL frames.
20	Brookhaven V206	A16 DE00-DFFF	60	RTDL Input Module P. Generates 8 RTDL frames.
21	(Reserved)			(2nd VME Bus Bridge – If Needed)

2.2 TIMING GATE CONNECTIONS

Most of the internal, or "intra-system" connections are between the timing gate trigger modules (V124S) and the event link input modules (V101S). Some gates trigger other system components such as the RTDL encoder module (V105S), or the ring RF frequency measurement counter.

The table below shows the connections between the timing gate trigger modules (V124S) and the hardware event input modules (V101S).

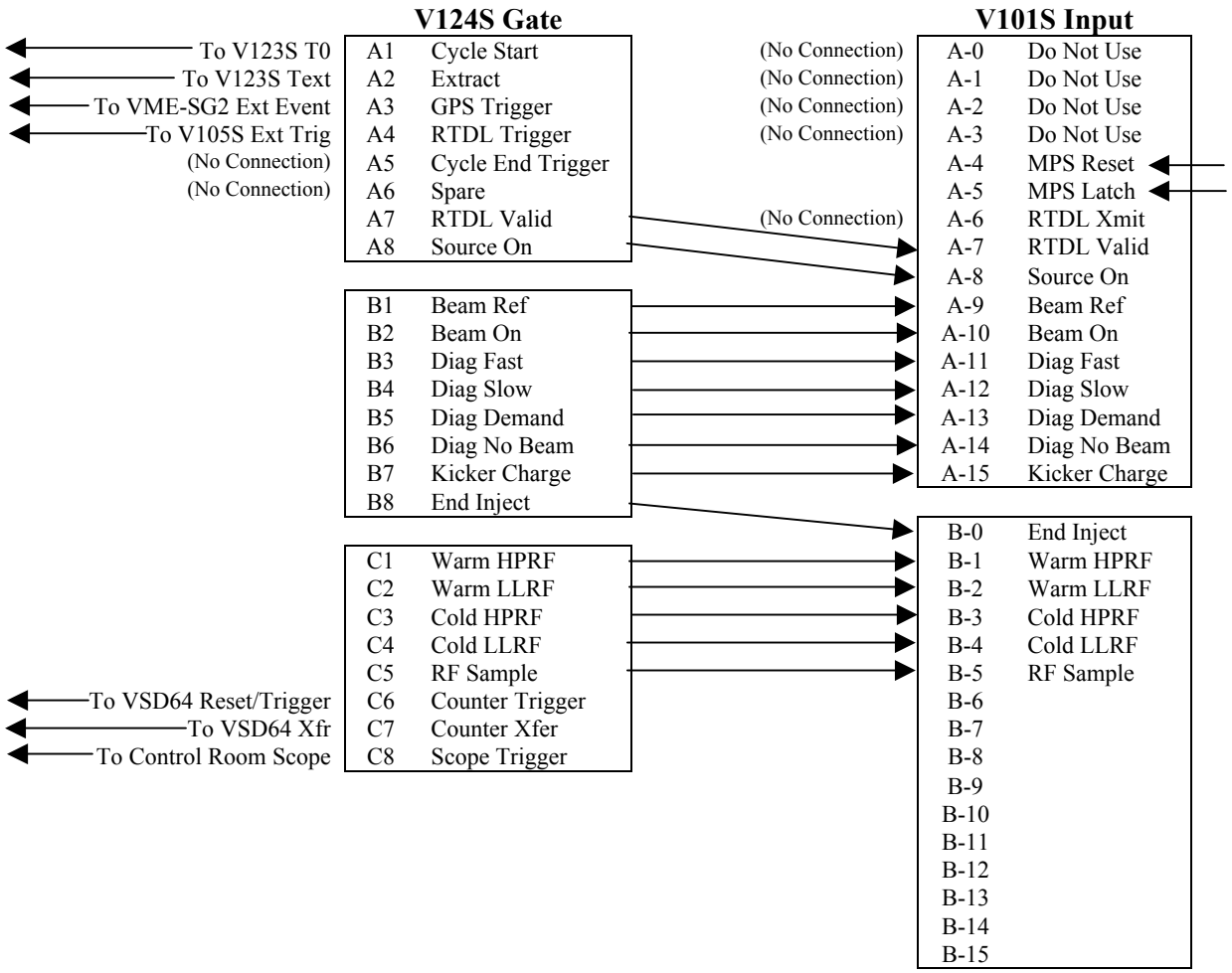


Figure 6
Timing Gate/Event Trigger Connections

2.3 MPS CONNECTIONS

The MPS has two connections, one for the MPS "Auto-Reset" signal and one for the MPS "Latched" signal, that go directly into two V101S channels. These connections come from the MPS "Trigger Control Chassis" and will cause an event to be broadcast on the event link whenever an MPS fault occurs. In addition, the MPS "Latched" status signal is connected to the timing master IOC's utility module via the binary I/O connector. The timing master IOC monitors this signal and disables delivery of the "Beam-On" event if MPS is faulted. The timing master IOC can also drop MPS itself, if it detects a condition that would indicate that beam should not be delivered (e.g. event link errors). This connection is also through the utility module's binary I/O connector.

The MPS connections into the timing master IOC's utility module are shown below:

Figure 7
MPS Status Input To Timing Master Utility Module

Note: This picture only shows the input connection from the MPS "Latched" signal. Once we get the output connection defined, this figure should be replaced with a diagram showing how the connections are made. Once we get the "Rev-D" versions of the utility module, we should also include some pictures showing how to strap the I/O jumpers for a timing master IOC.

2.4 OPERATOR SCREENS

The top level operator screen for the timing system is in the file:

```
$IOCTOP/timingMaster/production/opi/TimingMaster.edl
```

It is shown below in Figure 8.

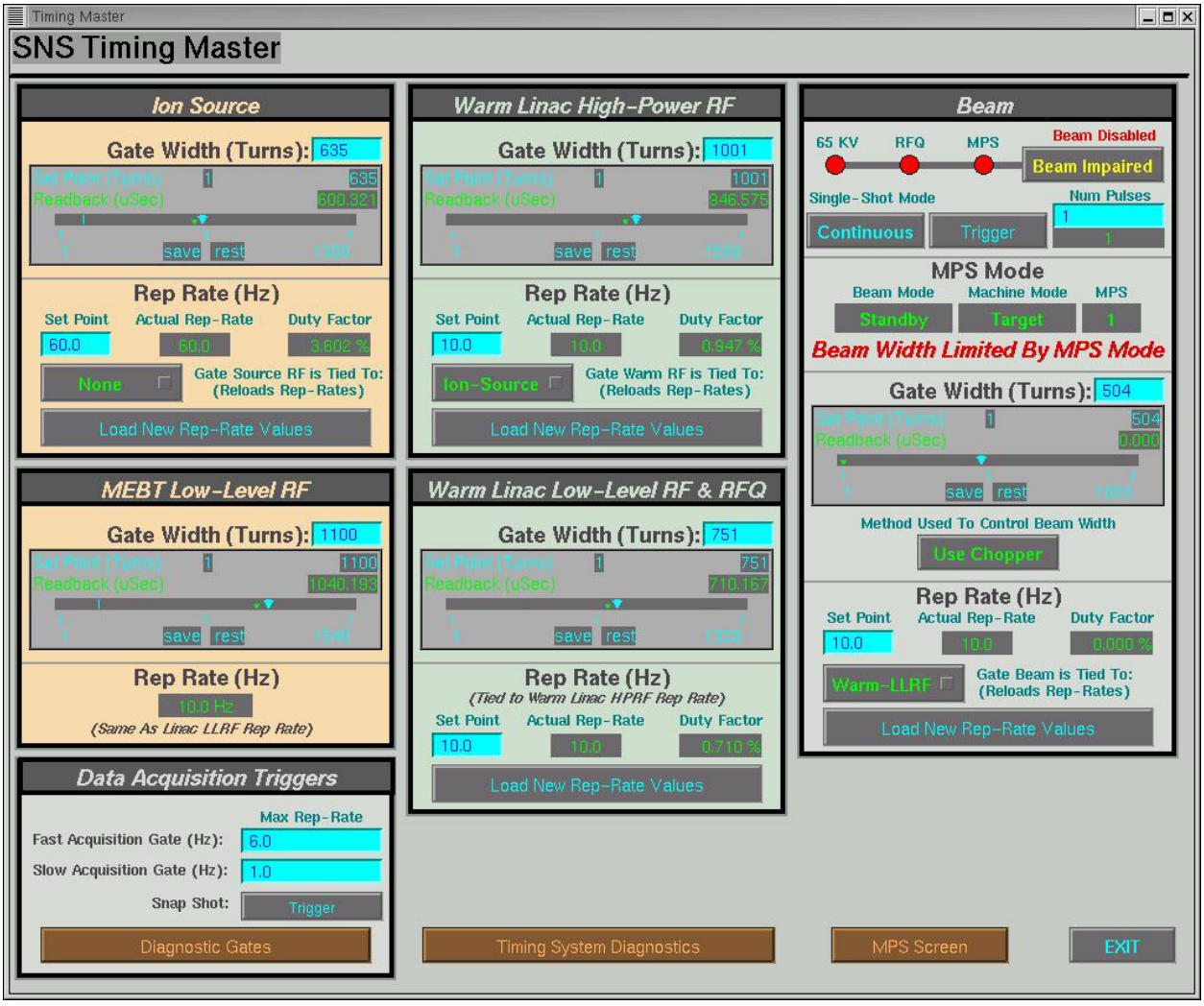


Figure 8
Top-Level Timing Screen

The screen is divided into sections for the "Front End" (including the ion source and the MEBT buncher RF), the "Warm (normal conducting) Linac" (including the high-power and low-level RF for the RFQ, DTL, and CCL), and a final section for the beam timing.

2.4.1 Ion Source Controls

The first section of the Timing Master screen, in the upper left corner of the screen, controls the timing gate for the ion source. Parameters that can be controlled are the gate's width, rep-rate, and dependent gate. The ion source gate always ends at the "End Inject" point of the machine cycle (see section 1.1.1 for a discussion of the machine cycle timeline). Therefore, adjusting the gate width will also

adjust its delay from the start of the cycle. The slider widget displays the delay in both turns and microseconds.

The rep-rate may be set to any value from 0 to 60 Hertz in 0.1 Hertz increments. The light blue box on the left is where the desired rep-rate is set. The dark gray box in the middle displays the rep-rate currently in effect. The dark gray box to the right displays the "duty factor" (the product of the rep-rate and the width). Entering a new rep-rate in the setpoint box does not immediately change the actual rep-rate. Changing a gate's rep-rate is a two-step process. First you must enter the new rep-rate in the setpoint box. Then you must press the "Load New Rep-Rate Values" button. This will cause all the variable rep-rate gates to re-compute their patterns (see the general note on changing rep-rates in section 2.4.8 below).

Normally, the ion source gate runs unconstrained – since it is the originator of the beam. If needed, however, the menu selection button labled "Gate Source RF is Tied To" can be used to constrain the ion source gate to be coincident with either the Warm Linac HPRF gate or the Warm Linac LLRF gate. Note that changing the dependency will cause all the rep-rate patterns to be recomputed – the same as if you had pushed the "Load New Rep-Rate Values" button.

2.4.2 MEBT Low-Level RF Controls

Just below the ion source control section, we have the low-level RF gate controls for the MEBT bunchers. All four bunchers use the same gate. The only parameter that can be changed is the gate width. Like the ion source gate, the MEBT LLRF gate always ends at the "End Inject" point of the machine cycle. Therefore, adjusting the gate width will also adjust its delay from the start of the cycle. The slider widget displays the delay in both turns and microseconds.

The rep-rate is displayed for convenience. The MEBT LLRF gate rep-rate is always the same as the Warm Linac LLRF gate rep-rate.

2.4.3 Data Acquisition Trigger Controls

The timing system provides four data acquisition events that are mainly used by the diagnostics. There is a "Fast" gate, a "Slow" gate, an "On Demand" gate, and a "No Beam" gate. The rep-rates of the fast and slow gates can be adjusted, however they should normally be set to 6 Hz and 1 Hz respectively.

The "Fast" gate only fires on cycles that have beam gates. The "Slow" gate only fires on cycles that have "Fast" gates (and, by implication, beam gates). The "On Demand" gate only fires on cycles that have both "Fast" and "Slow" (and "Beam") gates. The "On Demand" gate is fired by pressing the "Trigger" button on the "Data Acquisition Triggers" section of the screen. The "No Beam" gate fires at the same rate as the "Fast" gate, but out of phase with the beam. Unlike the "Fast" gate, the "No Beam" gate will fire regardless of whether there is any beam scheduled for that cycle or not.

Currently, the data acquisition trigger gates are handled as special cases within the timing master sequencer program. Consequently, changing their rep-rates takes effect immediately (without having to push the "Load New Rep-Rate Values" button) and does not cause any of the other rep-rate patterns to be recomputed.

The fast and slow gates have "minimum separation" provisions based on their respective rep-rates. For example, if the fast gate rep-rate is 6 Hz, then the fast gate will only fire on a cycle with beam that is at least 166.7 milliseconds (6 Hz.) from the previous fast gate. The slow gate will only fire on the same cycle as a fast gate and that is at least 1 second from the previous slow gate. One of the consequences of this feature is that the data trigger gates may not always run at their nominal rep-rates, even though the beam rep-rate is greater than 6 Hz. For example, if the beam rep-rate is an even 10 Hz, then the distance between beam gates is 100 milliseconds. The minimum separation provision of 166.7 milliseconds on the fast gate would mean that it could only select every other beam pulse, so the actual rep-rate of the fast gate would only be 5 Hz.

The "Diagnostic Gates" related-display button at the bottom of this section allows you to bring up timing control screens for the individual diagnostic devices.

2.4.4 Warm Linac High-Power RF Controls

The top box in the middle section of the Timing Master screen controls the timing gate for the Warm Linac and RFQ high-power RF. Parameters that can be controlled are the gate's width, rep-rate, and dependent gate. Like the ion source gate, the Warm HPRF gate always ends at the "End Inject" point of the machine cycle (see section 1.1.1 for a discussion of the machine cycle timeline). Therefore, adjusting the gate width will also adjust its delay from the start of the cycle. The slider widget displays the delay in both turns and microseconds.

The rep-rate may be set to any value from 0 to 60 Hertz in 0.1 Hertz increments. The light blue box on the left is where the desired rep-rate is set. The dark gray box in the middle displays the rep-rate currently in effect. The dark gray box to the right displays the "duty factor" (the product of the rep-rate and the width). Entering a new rep-rate in the setpoint box does not immediately change the actual rep-rate. Changing a gate's rep-rate is a two-step process. First you must enter the new rep-rate in the setpoint box. Then you must press the "Load New Rep-Rate Values" button. This will cause all the variable rep-rate gates to re-compute their patterns (see the general note on changing rep-rates in section 2.4.8 below).

Normally, the Warm HPRF gate is dependent on the ion source gate. If needed, however, the menu selection button labled "Gate Warm RF is Tied To" can be used to run the Warm HPRF gate independently (for conditioning purposes, for example). Note that changing the Warm HPRF dependency will cause all the rep-rate patterns to be recomputed – the same as if you had pushed the "Load New Rep-Rate Values" button.

2.4.5 Warm Linac Low-Level RF and RFQ Controls

Just below the "Warm Linac High-Power RF" section, we have the controls for the Warm Linac low-level RF and the RFQ timing gates. The only parameters that can be controlled are the gate's width and rep-rate. Like the ion source gate, the Warm LLRF gate always ends at the "End Inject" point of the machine cycle (see section 1.1.1 for a discussion of the machine cycle timeline). Therefore, adjusting the gate width will also adjust its delay from the start of the cycle. The slider widget displays the delay in both turns and microseconds.

The rep-rate may be set to any value from 0 to 60 Hertz in 0.1 Hertz increments. The light blue box on the left is where the desired rep-rate is set. The dark gray box in the middle displays the rep-rate currently in effect. The dark gray box to the right displays the "duty factor" (the product of the rep-rate and the width). Entering a new rep-rate in the setpoint box does not immediately change the actual rep-rate. Changing a gate's rep-rate is a two-step process. First you must enter the new rep-rate in the setpoint box. Then you must press the "Load New Rep-Rate Values" button. This will cause all the variable rep-rate gates to re-compute their patterns (see the general note on changing rep-rates in section 2.4.8 below).

The Warm LLRF gate depends on and is coincident with the Warm HPRF gate. Consequently, the actual Warm LLRF rep-rate will never exceed the Warm HPRF rep-rate (you can't send RF if the power supply is off) regardless of what the setpoint requests.

2.4.6 Beam Gate Controls

The right-hand section of the Timing Master screen controls the beam gate. This is the gate that is sent to the LEBT chopper that lets beam out of the source and into the RFQ.

The button in the upper right corner enables or disables beam delivery. When beam delivery is disabled (impaired), the ion source continues to pulse at its normal rep-rate (to preserve thermal stability), but the LEBT chopper does not allow any beam into the RFQ. Also, the source gate is shifted in time so that it is not coincident with the RFQ gate. This is so that even if the LEBT chopper fails, no beam will be accelerated past the RFQ. The line with the colored dots on it leading to the "Beam On" button form a small "Run Permit" chain. If the 65 KV or RFQ power supplies are not enabled, or if MPS is not made up, you will not be permitted to turn the beam on.

Under the "Beam On" button, we have the "Single-Shot" controls. In "Continuous" Mode, beam is steadily delivered at the selected width and rep-rate. In "Single-Shot" mode, beam delivery is inhibited until the operator pushes the "Trigger" button. Pushing "Trigger" allows a single shot of beam out of the ion source, and then disables beam until the next "shot". A "Shot" may consist of more than one beam pulse. Setting a value greater than 1 in the "Num Pulses" box will cause the "Trigger" button to deliver that number of pulses – at the specified rep-rate – and then inhibit the beam again. The gray readback box underneath the "Num Pulses" setpoint counts up to the number requested so that you can see how the "shot" is progressing.

Under the "Single-Shot" section, there is an informational section about the state of the MPS system. The "Beam Mode", "Machine Mode", and MPS mode number are all displayed. The MPS mode number is derived from the beam and machine modes. This is the number that is broadcast on RTDL frame 5. A message is displayed in this section (as shown above in Figure 8) if the "Beam Mode" constrains the beam width to be smaller than the value requested.

Under the "MPS Mode" section, we have the beam width section. Like the source and RF gates, the beam gate always ends at the "End Inject" point of the machine cycle (see section 1.1.1 for a discussion of the machine cycle timeline). Therefore, adjusting the gate width will also adjust its delay from the start of the cycle. The slider widget displays the delay in both turns and microseconds. The "turns" value is the setpoint and the "microseconds" value is the readback. The readback value reflects the actual width – including any abridgement by the MPS system.

The button labeled "Method Used to Control Beam Width" is a temporary measure until the LEBT chopper can be made to work reliably. If the LEBT chopper is working, then this button should have "Use Chopper" selected. If the LEBT chopper is not working, then the "Source Delay" method should be selected. The "Source Delay" method works by delaying the ion source gate such that it only intersects the RFQ RF gate for the amount of time specified by the beam width.

Note: **IF THE LEBT CHOPPER IS NOT WORKING, DO NOT SELECT THE "USE CHOPPER" METHOD.** Doing so could bypass normal MPS protections and allow too much beam into the accelerator.

The rep-rate may be set to any value from 0 to 60 Hertz in 0.1 Hertz increments. The light blue box on the left is where the desired rep-rate is set. The dark gray box in the middle displays the rep-rate currently in effect. The dark gray box to the right displays the "duty factor" (the product of the rep-rate and the width). Entering a new rep-rate in the setpoint box does not immediately change the actual rep-rate. Changing a gate's rep-rate is a two-step process. First you must enter the new rep-rate in the setpoint box. Then you must press the "Load New Rep-Rate Values" button. This will cause all the variable rep-rate gates to re-compute their patterns (see the general note on changing rep-rates in section 2.4.8 below).

Normally, the beam gate is dependent on the Warm LLRF gate (at least until we get the super-conducting accelerator built). If needed, however, the menu selection button labled "Gate Beam is Tied To" can be used to run the beam gate independently, or to make it depend on the ion source gate (you may want to do this if you want to run beam through the RFQ before the warm linac is ready). Note that changing the beam gate dependency will cause all the rep-rate patterns to be recomputed – the same as if you had pushed the "Load New Rep-Rate Values" button.

2.4.7 Other Related Displays

At the bottom of the Timing Master screen, there are buttons for bringing up related displays for timing system diagnostic/expert screens, and the MPS main screen. The timing system expert screens are described in subsequent sections of this document.

2.4.8 A General Note on Changing Rep-Rates

As mentioned numerous times in the preceding sections, changing a gate's rep-rate is a two-step process. First you must change the rep-rate setpoint, and then you must press one of the "Load New Rep-

Rate Values" buttons. It actually does not matter which one of the many "Load New Rep-Rate Values" buttons you press, they all have the same effect of causing all the variable rep-rate gates to recomputed their patterns. This means that you could change several rep-rate setpoints, and then make them all take effect by pressing a single "Load New Rep-Rate Values" button.

Each gate that has a variable rep-rate has a "rep-rate pattern" associated with it. This pattern is a 600-entry byte array that indicates which cycles during the 10-second "Super Cycle" this gate will be allowed to fire on. Since there can be dependencies between timing gates, the timing master must make sure that a gate's rep-rate pattern matches up with the rep-rate pattern of its dependent gate. Most of the time this is a rather quick and easy computation. However, if the desired rep-rates are not even divisors of 60, and are not even multiples of each other, it could take a while to sort the dependencies out. In some particularly difficult cases, the process could take up to 10 or 20 seconds.

Whenever a new set of rep-rate patterns is being computed, the Timing Master screen displays a message in the upper right-hand corner (in the "title bar") to let you know that it is working on the new pattern computation.

Figure 9
Message While Computing Rep-Rate Patterns

You should not attempt to change any rep-rates or dependencies while the timing master is setting new rep-rate values.

When the pattern computation has been completed, the Timing Master screen displays another message to let you know that the computation has completed and that a new set of rep-rate patterns are now in effect.

Figure 10
Message When Rep-Rate Computation is Complete

This message remains displayed for about ten seconds after the rep-rate pattern computation completes.

Changing a gate's dependencies will also, of necessity, require that the gate's rep-rate pattern be recomputed – along with the rep-rate patterns of any gates that depend on the changed gate. Therefore, the "Gate Dependency" menu selection buttons also act as "Load New Rep-Rate Values" buttons. Changing a gate's dependency will also cause all the variable rep-rate patterns to be re-computed.

Note that it is possible to create "circular dependencies". For example, you could specify that the ion source gate depends on the Warm LLRF gate and the Warm HPRF gate depends on the ion source gate. Since the Warm LLRF gate always depends on the Warm HPRF gate, you have a circular dependency. In such a case, the order of computation – and therefore the values of the rep-rate patterns – is indeterminate.

The rep-rate pattern computation process is described in more detail in section 8.5.

2.5 EPICS SOFTWARE

The EPICS software for the timing master IOC consists of:

- The timing master sequencer program.
- EPICS databases for monitoring and controlling the timing events and RTDL frames.
- SNL programs for producing "scope" displays of the timing gate delays and widths.

The software common to most of the timing master systems is contained in the directory:

```
$IOCTOP/timingMaster/production/timingLib
```

Software specific to individual timing masters is found in the timing master's application directory (e.g. timingMasterApp, devApp, lanlApp, jlabApp, etc.)

2.5.1 Timing Master Sequencer Program

The timing master sequencer program is found in the timing master application source directory:

```
$IOCTOP/timingMaster/production/timingMasterApp/src/timingMaster.c
```

Other timing masters will have their own copy of "timingMaster.c" in their own application directories.

The timing master sequencer program is started at boot time via the command:

```
start_timing
```

which is called in the "st.cmd" file after iocInit.

The function of the timing master sequencer program is to load and transmit the RTDL frames for the next cycle and "arm" the gates for any variable rep-rate events that are scheduled for the next cycle. It is a high-priority task that wakes up every cycle at about 100 turns past the Extract event. This allows us to send the RTDL frames for the next cycle about 10 milliseconds before the start of the next cycle (at 60 Hz). If the timing system runs at 120 Hz, the lead time is decreased to about 2 milliseconds.

The "start_timing" routine initializes the timing master IOC software. It performs the following actions:

- Initializes the tables used to compute the 24-bit CRC value that is sent as the last frame of the RTDL.
- Opens driver handles for the various VME cards it communicates with and does any additional hardware setup not already performed by the various driver initialization routines.
- Initializes the RTDL input channels for each of the RTDL frames it will be generating
- Initializes the event link input channels for each of the hardware events it will be generating and maps the V101 hardware event numbers into SNS hardware event numbers.
- Creates the high-priority timing sequencer task, "timingSeq".

When the timing master sequencer task is created, it performs some additional initialization and then enters its main processing loop. Most of the additional initialization involves the variable rep-rate system. It makes sure that the gates generating the Beam-On and Kicker-Charge events were successfully set up

as variable rep-rate gates and that they are "Soft Triggered", so that the sequencer task itself can make the final decision about whether or not they fire.

Just before entering its main loop, the task calls the "v124sSetNewRepRates" routine. This begins the process of computing the initial rep-rate patterns for the variable rep-rate gates (a process that requires a running timing master sequencer to complete). For more details about variable rep-rates, see section 8.5.

At the start of the main loop, the timing master sequencer calls "v124sWaitTrigger" to wait for the termination interrupt of a V124S gate (the "Cycle-End" gate) that is set up to fire 100 turns after the end of the "Time-Critical" section of the machine cycle (i.e. 100 turns after the "Extract" event). Once the "Cycle-End" gate fires, the task reads the time of the last Cycle-Start from the GPS system, increments the current cycle number, and obtains information about the MPS system and whether or not we can deliver beam this cycle. After that, the loop is divided into two parts that handle the event link and the RTDL for the next cycle.

2.5.1.1 Event Link Handling

The event link handling comes first. The timing master sequencer task "arms" any variable rep-rate gates scheduled to fire after the next Cycle-Start by calling the "v124sTriggerVariableGates" routine. This routine will not "arm" any "Soft Triggered" gates (such as Beam-On or Kicker-Charge), but will set the "trigger" field in their rep-rate information structures to indicate whether or not they are scheduled for the next cycle.

2.5.1.1.1 Special Case Variable Rep-Rate Gates

There are four variable rep-rate gates that are handled as special cases in the timing master sequencer task:

- **Beam-On:** The gate that generates the Beam-On event will only be armed to fire if 1) it is scheduled for this next cycle, 2) beam is enabled (MPS made up, power supplies on, not in single-shot mode, beam enable turned on…), and 3) the Kicker-Charge event fired on the last cycle.
- **Kicker-Charge:** The gate that generates the Kicker-Charge event will only be armed to fire if 1) it is scheduled for this next cycle, and 2) beam is enabled (MPS made up, power supplies on, not in single-shot mode, beam enable turned on…).
- **Fast-Diagnostic:** The Fast and Slow Diagnostic gates are even more special cases in that they don't even have variable rep-rate gensub records (see section 8.5.1). Their firing is determined completely by the timing master sequencer task. The gate that generates the Fast-Diagnostic event will only be armed to fire if 1) the Beam-On gate has been armed to fire on this cycle and 2) the length of time since the last time the Fast-Diagnostic event fired equals or exceeds the minimum separation based on the requested rep-rate for the Fast-Diagnostic event.
- **Slow-Diagnostic:** The Fast and Slow Diagnostic gates are even more special cases in that they don't even have variable rep-rate gensub records (see section 8.5.1). Their firing is determined completely by the timing master sequencer task. The gate that generates the Slow-Diagnostic event will only be armed to fire if 1) the Fast-Diagnostic gate has been armed to fire on this cycle and 2) the length of time since the last time the Slow-Diagnostic event fired equals or exceeds the minimum separation based on the requested rep-rate for the Slow-Diagnostic event.

2.5.1.1.2 Re-Arming Event Link Error Interrupts

After handling the variable rep-rate gates, the timing master sequencer re-enables error interrupts on the event link encoder module (V123S). The event link driver's interrupt service routine "throttles" event link errors to keep them from overwhelming the system and causing work queue panics. It does this by disabling interrupts whenever it gets an error interrupt and letting the timing master sequencer re-arm them. This effectively throttles the event link error rate down to 60 Hz.

The routine that re-arms the event link encoder's error interrupts (eventLinkArmErrors) returns a bitmask of any error conditions it encountered during the previous cycle. This bitmask is used in the second half of the sequencer loop when the RTDL "Veto" frame is constructed (see section 2.5.1.2.3).

2.5.1.1.3 Refreshing the Event Map Table

Every ten seconds, the timing master sequencer calls a V123S driver routine to refresh the in-core event mapping table from the hardware. This is the table that maps the hardware event numbers defined by the V101 channels and the event numbers broadcast on the event link (see sections 3 and 3.1.1). While the V123S card is on-line, you cannot read or write the event mapping table between Cycle Start and Extract without causing a bus error. In order to make the event map displayable, the V123S driver maintains an in-core buffer that can be accessed by the event link device support routines. Since the timing master sequencer is always called after the Extract event, this is a safe time to refresh the event map table from the hardware.

2.5.1.2 RTDL Handling

The second half of the timing master sequencer program's main loop set's up and broadcasts the RTDL frames for the next cycle.

2.5.1.2.1 Timestamp

The GPS interface is set up to "freeze" the current time when it receives a TTL signal from a gate that fires at Cycle-Start (see section 6). This is the time of the last Cycle-Start event. The timing master sequencer reads this time, adds 16⅔ milliseconds (for 60 Hz operation) and formats this into the RTDL frames containing the time of the next Cycle-Start event (see section 4.1.1).

2.5.1.2.2 Ring Revolution Period and Master Reference Generator

The ring revolution period is measured by counting the distance between two V124S gates separated by 16000 turns (see section 7). In addition to being broadcast on the RTDL, this value is used to compute the "Ring RF Compensation Delay" value that is written to the Master Reference Generator every cycle (see section 5). The timing master sequencer also reads the status register and the line frequency phase error from the Master Reference Generator. The phase error is broadcast in another RTDL frame and the status register is used in the "Last Cycle Veto" frame.

2.5.1.2.3 Veto Frame

The "Last Cycle Veto" frame is a collection of information from the MPS, the Master Reference Generator, the event link encoder, the ring revolution period measurement, and the beam flavor. Section 4.1.9 describes the composition of this frame.

The status of the "MPS Latched" signal is read from an input to the utility module (see section 2.3). The "Cycle-End" gate that triggers the timing master sequencer's main loop has its timestamp register set up to count "MPS Auto-Reset" events. This provides the "MPS Auto-Reset" input to the Veto frame.

The event link error status is obtained from the routine that re-arms the event link encoder's error interrupts (see sections 2.5.1.1.2 and 3.4).

2.5.1.2.4 Beam and RF Gate Widths

The desired gate widths for the Warm HPRF, Warm LLRF, Cold HPRF, Cold LLRF, and beam gates are maintained in vxWorks global variables by the databases that set them. The timing master sequencer task reads these variables and loads their values into the appropriate RTDL frames.

2.5.1.2.5 IOC Reset Frame

T.B.S.

2.5.1.2.6 CRC Frame

The last frame sent on the RTDL is a 24-bit CRC of the preceding frames and frame numbers. Details of this computation are given in [7].

2.5.2 Databases

Most of the databases used by the timing master IOC reside in the directory:

```
$IOCTOP/timingMaster/production/timingLib/Db
```

and are installed in the directory:

```
$IOCTOP/timingMaster/production/db
```

These databases are all template files and are generally expanded at build time by a substitution file in the timing master application directories "Db" file. The templates (and their macros) are:

timingMaster.template

This template contains miscellaneous records used the timing master IOC and timing master sequencer program. Records in this template perform functions such as:

- Broadcasting SNS software events on the event link
- Keeping track of the status of the variable rep-rate pattern computation process (see section 8.5.4)
- VxWorks variable support for various timing master functions such as the measured ring revolution period.

Macros:

S = System Name
N = System Instance (blank for the timing master, 2 for the backup)

The above macros are used to construct the PV names, which are of the form:

$(S)_Tim:<device>$(N):<signal>

eventLink.template

This template contains records that pertain to the event link generator module (V123S). See section 10.4 for a description of the specific signals.

Macros:

S = System Name
N = System Instance (blank for the timing master, 2 for the backup)

The above macros are used to construct the PV names, which are of the form:

$(S)_Tim:EventGen$(N):<signal>

event.template

This template contains records that define individual hardware event channels in the event input modules (V101). See section 10.5 for a description of the specific signals.

Macros:

S	= System Name
N	= System Instance (blank for the timing master, 2 for the backup)
EVENT	= The "V101 Number" (based on the V101 channel generating the event) for this event.
NAME	= The event name (displayed on the event input diagnostic screens)

The above macros are used to construct the PV names, which are of the form:

$(S)_Tim:Event$(N)_$(EVENT):<signal>

v206.template

This template contains records that pertain to the RTDL input card (V206) as a whole. See section 0 for a description of the specific signals.

Macros:

S	=	System Name
N	=	System Instance (blank for the timing master, 2 for the backup)
DI	=	The Device Instance (typically the V206 card designator (A,B,C,…)).
CD	=	The Card Number of the V206 (used in the VME address fields)

The above macros are used to construct the PV names, which are of the form:

$(S)_Tim:RTDLIn$(N)_$(DI):<signal>

rtdlInput.template

This template contains records that define individual RTDL input channels in the V206 modules. See section 10.8 for a description of the specific signals.

Macros:

S	= System Name
N	= System Instance (blank for the timing master, 2 for the backup)
DI	= The Device Instance (the V206 card designator (A,B,C,…)).
CD	= The Card Number of the V206 (used in the VME address fields).
CH	= V206 Channel Number (starting at 1)
NAME	= The RTDL frame name (displayed on the RTDL input module diagnostic screens)

The above macros are used to construct the PV names, which are of the form:

$(S)_Tim:RTDL$(N)_(DI)(CH):<signal>

Note that the RTDL frame number does not appear anywhere in this template (even though the name does). The timing master sequencer program assigns the RTDL frame numbers.

BeamControl.template

This template contains records that control the gates that generate beam-related events. The gates in this template generate the following events:

- Beam Reference (event 37)
- Beam On (event 36)
- Fast Diagnostic – Rep-rate only (event 47)
- Slow Diagnostic – Rep-rate only (event 46)
- On-Demand Diagnostic (event 45)
- Kicker Charge (event 40)

In addition, this template also contains the records for reading the MPS status from the MPS Master IOC and the states of the RFQ and 65 KV high voltage power supplies (used to determine whether or not we can turn beam on).

Macros:

S	=	System Name
N	=	System Instance (blank for the timing master, 2 for the backup)
BeamRefAddr	=	Complete VME address of the V124S gate that generates the Beam Reference event ("C*x* S*y*")
DiagDmdAddr	=	Complete VME address of the V124S gate that generates the On-Demand Diagnostic event ("C*x* S*y*")
Con	=	Card number for the V124S gate that generates the Beam On event
Gon	=	Gate (channel) number for the V124S gate that generates the Beam On event
Ckick	=	Card number for the V124S gate that generates the Kicker Charge event
Gkick	=	Gate (channel) number for the V124S gate that generates the Kicker Charge event
MaxTurns	=	Upper limit on the width of the beam gate (in turns)
Limit	=	Upper limit of the "Mostly Stable" region of the machine cycle where the beam and RF-related events occur (see Table 2 in section 3)

Four V124S gates are specified in this template. Those gates that do not have variable rep-rates (Beam Reference and On-Demand Diagnostic) have their VME addresses specified by a single macro. Those gates that do have variable rep-rates (Beam On and Kicker Charge) have their addresses separated into "card" and "gate" components for input into the variable rep-rate gensub record (see section 8.5.1). Note that the fast and slow diagnostic events do not have variable rep-rate gensub records because they are handled as special cases by the timing master sequencer task (see section 3.1.2).

SourceControl.template

This template contains records that control the gate that generates the Source-On event.

Macros:

S	= System Name
N	= System Instance (blank for the timing master, 2 for the backup)
CD	= The card number of the V124S gate used to generate the event
CH	= The channel number (starting at 1) of the V124S gate used to generate the event.
MaxRep	= Maximum Rep-Rate allowed.
Event	= Event that triggers the gate that triggers the Source-On event.
RevDly	= Number of turns to delay after the trigger event.
AutoReload	= Whether or not to automatically reload the gate's counters

The above macros are used to construct the PV names, which are of the form:

$(S)_Tim:Gate$(N)_SourceOn:<signal>

WarmHPRFControl.template

This template contains records that control the gate that generates the Warm Linac High-Power RF event.

Macros:

S	= System Name
N	= System Instance (blank for the timing master, 2 for the backup)
CD	= The card number of the V124S gate used to generate the event
CH	= The channel number (starting at 1) of the V124S gate used to generate the event.
MaxRep	= Maximum Rep-Rate allowed.
MaxTurns	= Maximum width (in turns) of the Warm HPRF gate.
Limit	= Upper limit of the "Mostly Stable" region of the machine cycle where the beam and RF-related events occur (see Table 2 in section 3)
Event	= Event that triggers the gate that triggers the Warm HPRF event.
AutoReload	= Whether or not to automatically reload the gate's counters

The above macros are used to construct the PV names, which are of the form:

$(S)_Tim:Gate$(N)_WarmHPRF:<signal>

WarmLLRFControl.template

This template contains records that control the gate that generates the Warm Linac Low-Level RF event.

Macros:

S	= System Name
N	= System Instance (blank for the timing master, 2 for the backup)
CD	= The card number of the V124S gate used to generate the event
CH	= The channel number (starting at 1) of the V124S gate used to generate the event.
MaxRep	= Maximum Rep-Rate allowed.
MaxTurns	= Maximum width (in turns) of the Warm LLRF gate.
Limit	= Upper limit of the "Mostly Stable" region of the machine cycle where the beam and RF-related events occur (see Table 2 in section 3)

Event = Event that triggers the gate that triggers the Warm HPRF event.
AutoReload = Whether or not to automatically reload the gate's counters

The above macros are used to construct the PV names, which are of the form:

$(S)_Tim:Gate$(N)_WarmLLRF:<signal>

llrfSampleEvent.template

This template contains records that control the gate that generates the Warm Linac Low-Level RF Sample event.

Macros:

S = System Name
N = System Instance (blank for the timing master, 2 for the backup)
CD = The card number of the V124S gate used to generate the event
CH = The channel number (starting at 1) of the V124S gate used to generate the event.

The above macros are used to construct the PV names, which are of the form:

$(S)_Tim:Gate$(N)_RFSample:<signal>

timingScopeSettings.template

This template contains records that set the start and stop times for the timing scope display (see section 2.5.3).

Macros:

S = System Name
SS = Sub-System Name
N = System Instance (blank for the timing master, 2 for the backup)
DI = Device Instance.

timingScope.template

This template contains records for displaying a single trace on the the timing scope display (see section 2.5.3).

Macros:

S = System Name
SS = Sub-System Name
N = System Instance (blank for the timing master, 2 for the backup)
DI = Device Instance.

The above macros are used to construct the PV names, which are of the form:

$(S)_$(SS):SCOPE$(N)_$(DI):<signal>

2.5.3 Scope Display SNL Progams

T.B.S.

3 EVENT LINK

The SNS Event Link is modeled after the RHIC Beam Synchronous Link [4]. It is a differential PECL, bi-phase mark encoded, serial link that transmits 8-bit event codes synchronized with the ring revolution period. An SNS event actually contains 12 bits. These are a start bit, an 8-bit frame number, a parity bit, and two stop bits. The format of an event frame is show below:

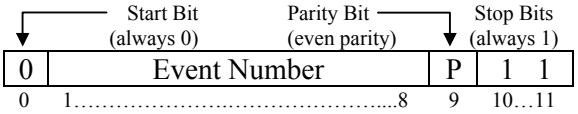


Figure 11
Event Frame Format

The frame is transmitted in "Big-Endian" format (i.e. most significant bit first).

Events are divided into two categories – hardware events and software events. As the name implies, hardware events are generated by the event link hardware. They have (or can have) very precise timing and are prioritized based on the hardware port that generated them. The three V123S-generated events have the highest priority. Events generated by the V101 card closest to the V123S have the next highest priority, with channel 0 having a higher priority than channel 1, channel 1 having a higher priority than channel 2, etc.

The hardware events are numbered from 0 to 63. This corresponds to the fact that there are four event input (V101) modules with sixteen inputs each. The event number broadcast on the event link, does not necessarily have to be the same as the number of the hardware event that generated it. The event encoder module (V123S) contains an event mapping table that determines which event number actually gets transmitted when a particular hardware event is triggered. This allows us to arrange event numbers in groups that make sense logically, and to determine their relative priorities based on the V101 channel that generates it.

In this document, we distinguish between the "SNS Event" (the event that is broadcast on the event link) and the "V101 Event" (the hardware event number based on the V101 channel). To compute the "V101 Event" number, take the V101 card (1,2,3, or 4), subtract one, multiply by 16, and add the channel number (0 – 15).

Software events are generated (again, as the name implies) by software running on the timing master IOC. They do not have precise timing and are prioritized on a "First-In-First-Out" basis. Software events are usually either informational (e.g. the rep-rate patterns have been changed) or signal some non-time critical activity (e.g. clear all MPS error counters). Software events are not allowed to occur during the "Time Critical" portion of the machine cycle between Cycle-Start and Extract (see section 1.1.1). If a software event is requested during this time, it is stored in a FIFO internal to the V123S module and not released until after the Extract event.

Software events are numbered from 64 to 255. Like the hardware events, software events are also mapped through the V123S module's event map. In order to avoid confusion, the SNS timing system makes this an "identity" mapping for all software events (note that this also implies that the hardware event mapping, while not identical, still maps within the range of hardware event numbers).

SNS events have several functions:

- They define the fixed portions of the SNS machine cycle (i.e. Cycle-Start, End-Inject, Extract, or RTDL-Valid).
- They set relative locations for those variable delay and width gates that need to be synchronized across the entire accelerator (e.g. RF gates, start of beam, or certain diagnostic pulses).

- They control the repetition rates of variable-rate gates (e.g. ion source, RF gates, or start of beam).

Most of the events used to define gates with variable rep-rates, widths, and/or delays occur fairly far in advance of the actual gate being generated. This situation is shown below in the more detailed SNS machine cycle timeline diagram:

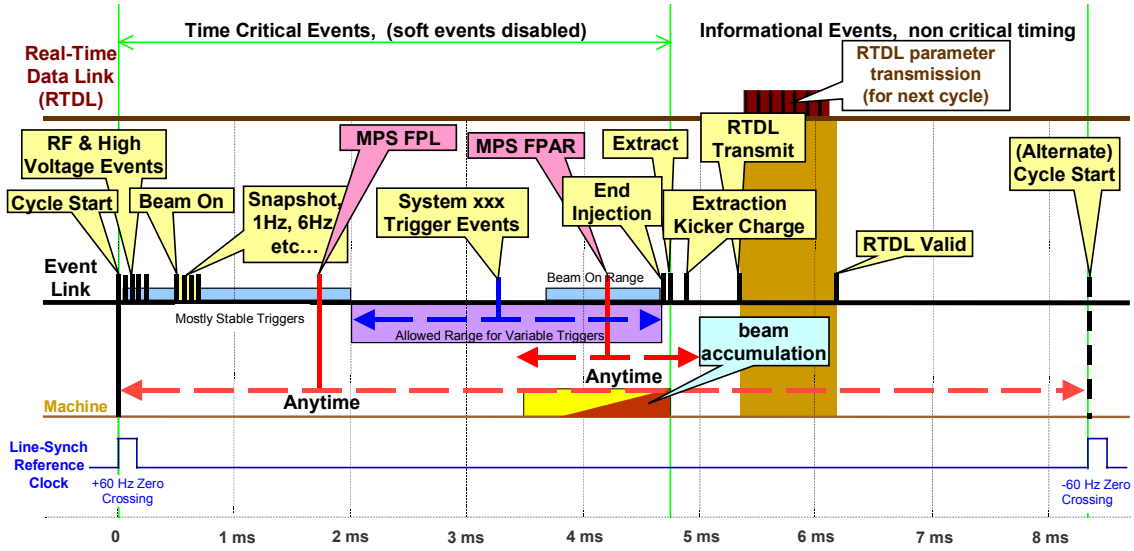


Figure 12
Detailed Machine Cycle Time Line

Notice that the variable-gate definition events (RF, High Voltage, Beam On, Diagnostic Triggers) all occur on the left hand side of the timeline, near Cycle-Start, in an area labeled "Mostly Stable Triggers." The gates they generate, however, all occur on the right hand side of the time-critical section of the timeline, in an area labeled "Beam On Range." This allows timing system client applications to define triggers or "pre-pulses" that can come up to about 2.5 milliseconds before the actual gate, if needed.

The label "Mostly Stable Triggers" means that while events can be moved around within this area, their positions will be stable during production beam delivery. One of the things to worry about in an event link system is "Event Jostle." Event Jostle occurs when two different events, defined by two different V101 input channels, attempt to fire at the same time. Since the event link is a serial link, both events cannot be broadcast simultaneously, and the lower priority event will have to wait until the higher priority event is finished. The length of the delay can be anywhere from 57 to 730 nanoseconds, depending on when the collision occurred and what the ring revolution period is. Note also that priority alone will not prevent your event from being jostled. If the lower priority event occurs slightly ahead of the higher priority event, the higher priority event will be delayed until the current event is finished. This is why soft events are locked out during the time-critical portion of the machine cycle.

The purple area labeled “Allowed Range for Variable Triggers”, with the “System xxx Trigger Events” box pointing to it. Is an area for system-specific single shot triggers events which will not interfere with the “Mostly Stable” events.

Note that the MPS trip events (Fast-Protect Auto-Reset and Fast-Protect Latch) can occur at any time during the machine cycle. These events have very high priority (their V101 event numbers are 4 and 5, respectively). An MPS trip event could jostle another time-critical event. However, if an MPS trip occurs, the machine cycle is hosed anyway.

The following table details the timing of the fixed portions of the machine cycle timeline. All times are specified in turns past cycle start.

Time-Critical Section	0-5050	Beam acceleration, accumulation, and extraction portion of the timeline. Software events are locked out during this time. There should be no event “jostle” during this period.
Mostly Stable Triggers	10-2110	Events for variable rep-rate and variable width beam and RF gates occur in this area. Although these events can move, the operational constraints of the accelerator should keep them from interfering with each other.
Variable Trigger Range	2111-5047	Single shot system-specific events occur during this period.
End Inject	5048	Ending time for all beam and RF gates.
Beam-On Range	2119-5048	This is the time period during which the gates triggered by events in the “Mostly Stable” region actually fire.
Kicker Charge	5062	Start charging extraction kicker for the next cycle.
Cycle-End	5150	This is the point, 100 turns after the end of the time-critical section, that the timing master sequencer computes the RTDL frames and enables the variable-rep-rate gates for the next cycle.

Table 2
Machine Cycle Timeline Definitions

3.1 SNS EVENT DEFINITIONS

3.1.1 Hardware Events

The following table lists the hardware events currently implemented in the SNS timing system. The first column gives the “V101 Event Number”. This is the number that would show up in the “Signal” number field of an EPICS record (see section 10.5). This is also the number that would be used in the “event” argument in calls to the event link driver software (see section 3.4). The “V101 Event Number” also determines the hardware priority of the event.

The “SNS Number” is the number of the event that is broadcast on the event link in response to a trigger of the corresponding V101 channel. The “Trigger Event” and “Delay from Trigger” fields refer to the V124S gate that generates this event. For events that occur in the “Mostly Stable” or “Variable Trigger” regions that generate gates in the “Beam On” range, the delay from the event to the start of the gate is given in the “Delay to Gate” column.

V101 Num (Priority)	SNS Event Num	Event Name	Trigger Event	Delay From Trigger	Delay To Gate	Comments
0	0	----	----	----	----	Not Used
1	39	Extract	1	5050 Turns	----	60 Hz. Generated by V123S. Defines the end of the time-critical section of the timeline. Re-enables soft event delivery.
2	1	Cycle-Start	63	1 Sub-Rev	----	60 Hz. Generated by V123S. Defines the start of the time-critical section of the timeline.
3	63	Pre-Pulse	----	----	----	60 Hz. Generated by V123S. Triggered from master reference generator. Disables soft event delivery until the next Extract cycle. Will be eliminated in next V123S Rev.
4	3	MPS-Reset	----	----	----	Single-Shot. Triggered by MPS Auto-Reset signal.
5	4	MPS-Latch	----	----	----	Single-Shot. Triggered by MPS Latch signal.
6	43	RTDL-Xmit	----	----	----	60 Hz. Triggered by timing master sequencer after all the RTDL frames for the next cycle have been loaded. Triggers the gate that starts the RTDL transmission.
7	44	RTDL-Valid	43	6 × (Num RTDL Frames) + 1		60 Hz. Signals the end of the RTDL transmission.
8	27	Source-On	1	2 Turns	3747 to 5046 Turns	Variable Rep-Rate. Controls ion source rep-rate only. Event location remains fixed. Gate width and delay are set in the MPS Master IOC (which generates the Ion Source Gate). Can be independent, or made coincident with the Warm-HPRF or Warm-LLRF events.
9	37	Beam-Ref	1	1050-2109 Turns	2939 Turns	60 Hz. Event controls where the start of beam occurs.
10	36	Beam-On	37	2 Turns	2937 Turns	Variable Rep-Rate. Event controls the beam rep-rate. Can be independent, or made coincident with Source-On, or Warm-LLRF. Event will be inhibited if the Kicker-Charge event did not fire on the previous cycle.
11	47	Diag-Fast	36	2 Turns	2935 Turns	6 Hz. (Nominal) Always occurs on same cycle as Beam-On event. Actual rep-rate is determined by Beam-On rep-rate and the fact that the timing master sequencer guarantees that the there will always be at least 166.7 milliseconds between pulses. So, for example, if the Beam-On rep-rate is 10 Hz, the Diag-Fast rep-rate will actually be 5 Hz (since at 10 Hz there is only 100 milliseconds between pulses). Nominal rep-rate can be adjusted.
12	46	Diag-Slow	47	2 Turns	2933 Turns	1 Hz. (Nominal) Always occurs on same cycle as Diag-Fast event. Actual rep-rate is determined by the actual Diag-Fast rep-rate and be the fact that the timing master sequencer guarantees that there will always be at least one second between pulses.

V101 Num (Priority)	SNS Event Num	Event Name	Trigger Event	Delay From Trigger	Delay To Gate	Comments
						Nominal rep-rate can be adjusted.
13	45	Diag-Demand	46	2 Turns	2931 Turns	Single-Shot. Always occurs on same cycle as Diag-Slow and Diag-Fast events.
14	48	Diag-No-Beam	39	1 Turns	2931 Turns	6 Hz. Not necessarily coincident with anything. This is a 6 Hz. event that can be used to test diagnostic triggers when there is no beam. Trigger and delay put the gate outside the beam acceleration/accumulation/extraction part of the machine cycle, so you are guaranteed not to have any beam. Rep-rate is the same as the Diag-Fast nominal rep-rate.
15	40	Kicker-Charge	39	12 Turns	1 SubRev	Variable Rep-Rate Instructs the extraction kickers to begin charging. Must occur at least 13 milliseconds before extraction. Consequently, this gate must fire after the Extract event of the previous cycle. Rep-rate tied to Beam-On rep-rate. Occurs one cycle before Beam-On event. If beam is inhibited, this event will not be generated. Beam-On event will be inhibited if this event did not fire on the previous cycle.
16	38	End-Inject	1	5048 Turns	----	60 Hz. Occurs 2 cycles before Extract. This event signals the fixed end point for all beam and RF gates.
17	28	Warm-HPRF	1	530-2109 Turns	2939 Turns	Variable Rep-Rate. Controls the warm linac high-power RF gate width and rep-rate. Can be independent, or made coincident with the Source-On event.
18	29	Warm-LLRF	1	790-2109 Turns	2939 Turns	Variable Rep-Rate. Controls the warm linac low-level RF gate width and rep-rate. Always occurs on the same cycle as the Warm-HPRF event.
19	30	Cold-HPRF	1	10-2109 Turns	2939 Turns	Variable Rep-Rate. Controls the super-conducting linac high-power RF gate width and rep-rate. Can be independent, or made coincident with the Source-On event or the Warm-HPRF event.
20	31	Cold-LLRF	1	270-2109 Turns	2939 Turns	Variable Rep-Rate. Controls the super-conducting linac low-level RF gate width and rep-rate. Always occurs on the same cycle as the Cold-HPRF event.
21	50	RF-Sample	29	1329-4249 Turns	608 Turns	Single-Shot. Triggers RF diagnostic gate. Always occurs on the same cycle as a Warm-LLRF event. Gate timing can be set to occur anywhere from 1000 turns before the start of the Warm-LLRF gate to 1900 turns after the start of the Warm-LLRF gate.

Table 3
Hardware Event Definitions

3.1.2 The Beam-Related Event Cluster

The "Beam/Diagnostic Event Cluster" is worth mentioning at this point. This is a cluster of five events that are locked to each other and travel together as a fixed group. The cluster begins with the "Beam Ref" event. This is the only event that has a variable delay. It runs at 60 Hz and occurs 2939 turns before the start of the actual beam gate. It can be used to produce a diagnostic "fiducial" gate that tells you where beam would turn on if it were scheduled for this cycle.

The "Beam-On" event is triggered 2 turns after the "Beam Ref" event. This is the event that actually generates the "Beam Gate" (i.e. the gate that goes to the LEBT chopper and lets the beam out of the source). This event is subject to the beam rep-rate set in the Timing Master EDM screen. This event is also subject to the "Single-Shot" and "Beam Off" buttons on the Timing Master EDM screen. If beam is inhibited for any reason (e.g. MPS trip, or "Beam Off" selected), the Beam-On event will not be generated. The Beam-On event will also be inhibited if the Kicker-Charge event did not fire on the previous cycle. This provision allows you to change rep-rate patterns in the middle of a super cycle without accelerating beam before the extraction kickers have charged.

The three diagnostic triggers, Diag-Fast, Diag-Slow, and Diag-Demand, daisy-chain off of the Beam-On event. Diag-Fast is tied to Beam-On, Diag-Slow is tied to Diag-Fast, and Diag-Demand is tied to Diag-Slow. Diag-Fast and Diag-Slow have "minimum separation" provisions. The Diag-Fast event will only fire on a cycle with a Beam-On event that is at least 166.7 milliseconds (6 Hz.) from the previous Diag-Fast event. Diag-Slow will only fire on a cycle with a Diag-Fast event that is at least 1 second from the previous Diag-Slow event. One of the consequences of this feature is that the diagnostic gates may not always run at their nominal rep-rates, even though the Beam-On rep-rate is greater than 6 Hz. For example, if the Beam-On rep-rate is an even 10 Hz, then there are only 100 milliseconds between Beam-On events. The minimum separation provision of 166.7 milliseconds on the Diag-Fast event would mean that it could only select every other Beam-On pulse, so the actual rep-rate of Diag-Fast would only be 5 Hz.

The behavior described above, is not the behavior specified in the timing system final design review. The above implementation was done to accommodate "dumb" diagnostics that only had a single trigger and no way of knowing whether or not there was any beam associated with that trigger. Thus, the philosophy was "No Beam, No Trigger". Once the diagnostics start being able to listen to the event link and the RTDL, the diagnostic trigger event mechanism may need to be reconsidered.

The following figure illustrates the various relationships between which events trigger which other events, and when those events occur.

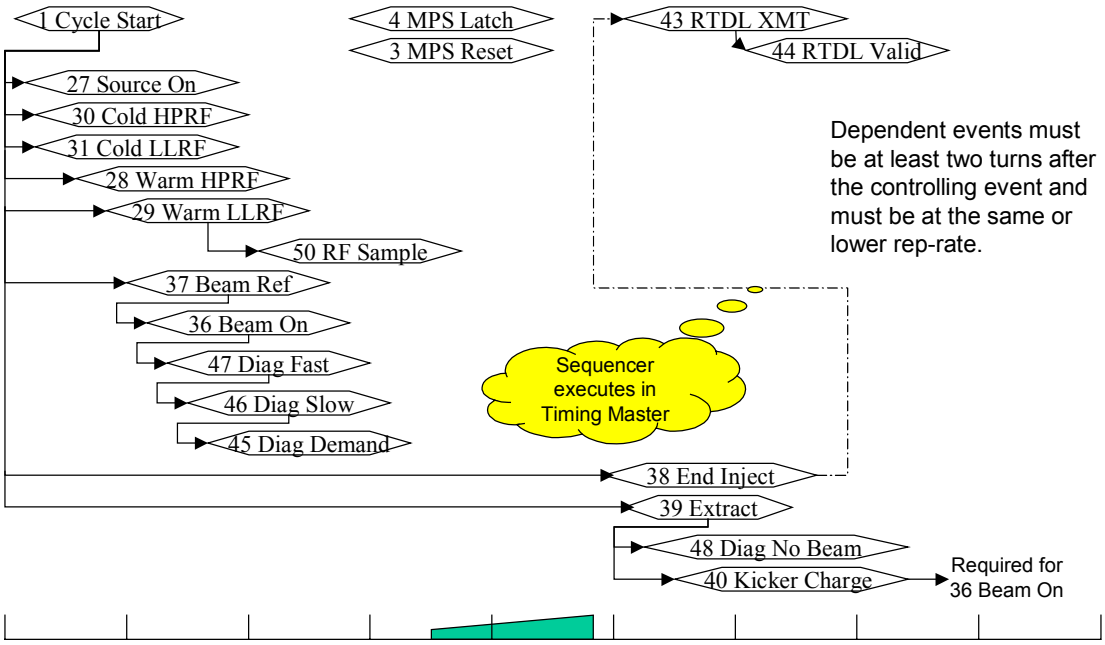


Figure 13
Hardware Event Relationships

3.1.3 Software Events

The following table lists the software events currently in use by the SNS timing system. All software events must ultimately be generated by the timing master IOC (because that's where the event generator lives). Note that for software events there is no difference between the "V101 Event Number" and the "SNS Event Number".

Event Num	Event Name	Comments
251	Compute-Rep-Rate	Signals that a new rep-rate pattern computation has begun in the timing master IOC.
252	New-Rep-Rate	Signals that new rep-rate patterns have been set in the timing master IOC.
253	MPS-Error-Reset	Reset error counters on all MPS chassis.
254	Util-Error-Reset	Reset error counters on all SNS Utility Modules.

Table 4
Software Event Definitions

3.2 HARDWARE

The SNS event link consists of up to five VME modules connected by a P2 backplane. The five modules are an event link encoder (V123S) and up to four event link input (V101) modules. These five modules can be logically treated as a single "Event Link Generator" module. The V123S module maps out address and interrupt vector space for itself and the four input modules. The P2 backplane handles all the communication between the V123S and the V101 modules.

3.2.1 Connections

A P2 backplane connector must be installed in the VME slots containing the V123S (slot 6) and the V101 (slots 7-10) modules. The backplane connector is installed on the upper pins of the P2 connectors as shown below:

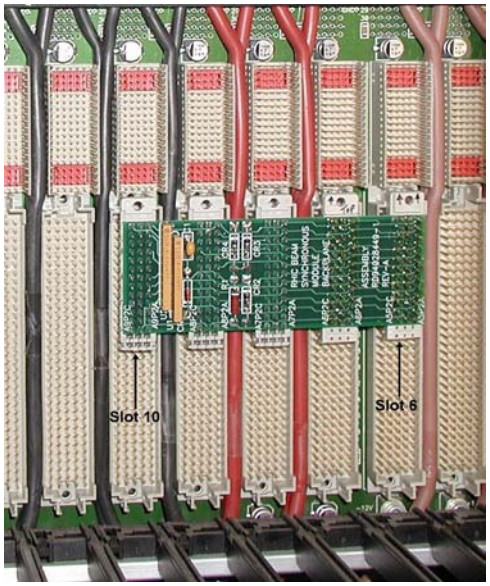


Figure 14
Event Link Backplane

3.2.1.1 Event Encoder (V123S) Module

The current V123S module has four input connections and one output. The input connections include the 32Xfrev RF clock input and three TTL event trigger inputs. The output connection is a twinaxial BNC connector that connects the event link output to the event link distribution panel.

The "RF CLOCK IN" port is an SMA input terminated into 50 ohms. The input expects a 10 dBm sine wave, 1 volt peak, or 2 volt peak-to-peak signal. This input can also be a PECL square wave.

The three event trigger inputs are labeled "PP", "T0", and "TEXT". PP is the "Pre-Pulse" event. It is triggered from the "Cycle Start" output of the master reference generator (see section 5.1). On the current V123S implementation, it is the Pre-Pulse event that locks out software events (instead of the Cycle-Start event). Software events are locked out until the next "Extract" event. The Pre-Pulse event (and the PP input) will be eliminated in the next revision of the V123S.

The "T0" input generates the "Cycle Start" event. It is triggered from a V124S gate that triggers shortly after the "Pre-Pulse" event.

The "TEXT" input generates the "Extract" event. It is triggered from a V124S gate that triggers 5050 turns after the Cycle-Start event. Once the Extract event fires, software events are re-enabled. Any software events that were queued during the Pre-Pulse to Extract interval will be released from the FIFO at this time.

Figure 15 (below) shows the various connections to the V123S module. Connections from the V124S trigger gates are shown in Figure 6 (above).

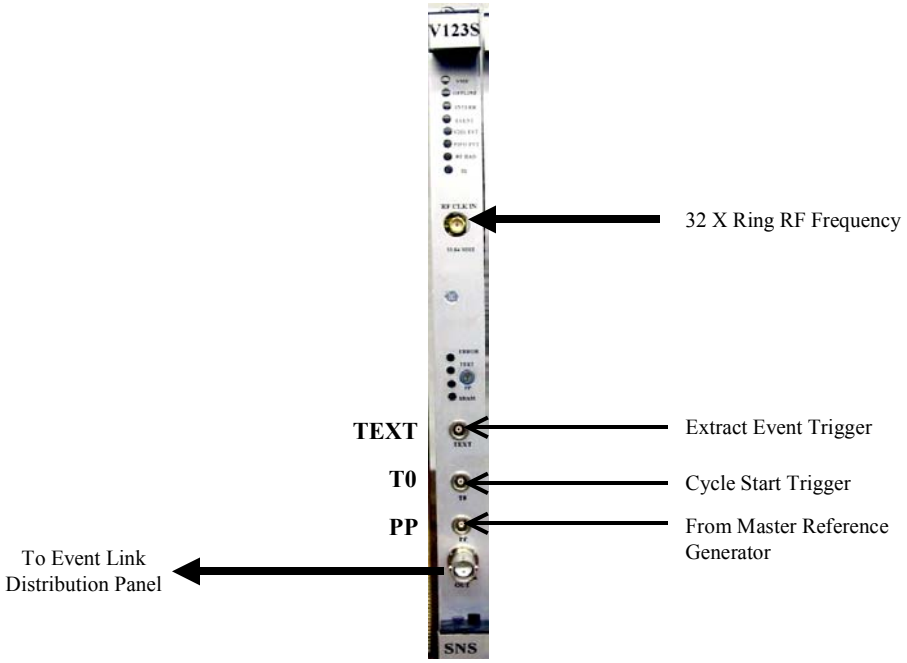


Figure 15
Connections to Event Link Encoder (V123S)

3.2.1.2 Event Input (V101) Modules

In general, the event trigger inputs on the V101 input modules are connected to V124S trigger module output gates. Figure 6 in section 2.2 shows the connections between the V101 inputs and the V124S outputs. The exceptions are the MPS "Auto Reset" and "Latched" events, which come from the MPS system, and the "RTDL Xmit" event input, which is software triggered from the timing master sequencer program.

The first four inputs on the first V101 card are actually duplicates of the event inputs on the V123S module. Channel 0 is the "Null" event (with no corresponding hardware input on the V123S). Channel one corresponds to the "TEXT" input on the V123S, channel two corresponds to the T0 input on the V123S, and channel three corresponds to the "PP" input on the V123S. These four V101 inputs should be disabled. Any signal on these inputs will conflict with the V123S inputs and cause errors.

3.2.2 Jumper Settings

The VME A16 base address for the entire event link system (encoder plus input modules) is set via four jumpers located in the middle of the V123S card (see Figure 16 below). The jumpers set base address lines A12 - A15 (4K boundary). With the board standing upright (as shown below), the high-order bit is

at the bottom and the low order bit is toward the top. If a jumper is present, it represents a logical “0”. If the jumper is removed, it represents a logical “1”.

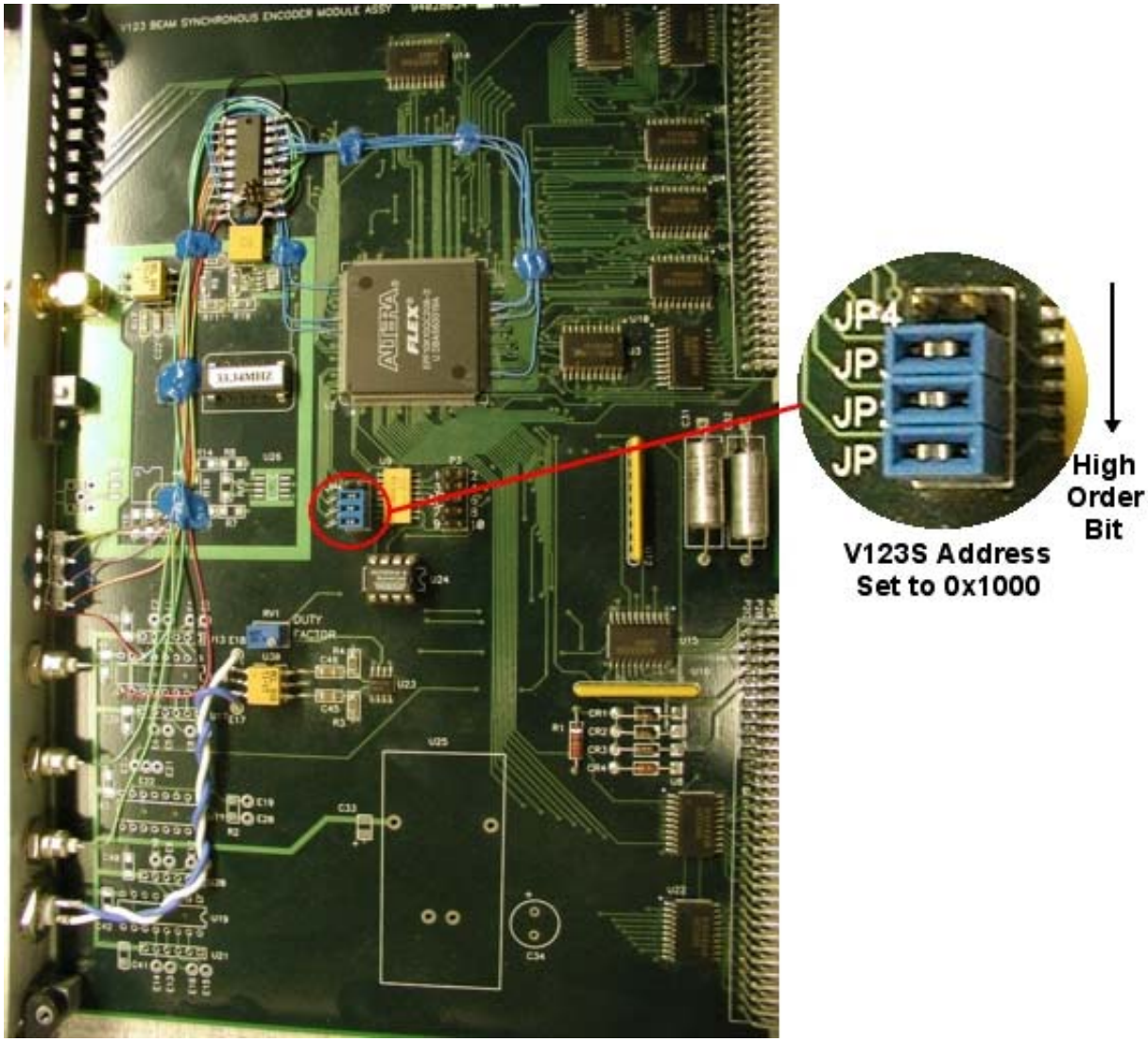


Figure 16
V123S Address Selection

3.3 DIAGNOSTIC SCREENS

Only diagnostic “expert” screens are available for the event link system. The diagnostic screens can be obtained by starting at the Timing Master EDM screen and selecting the “Timing System Diagnostics” related display button. From the Timing System Diagnostic screen, select the “EVENT Encoder (V123S) Screen” related display button. This will bring up the screen shown below:

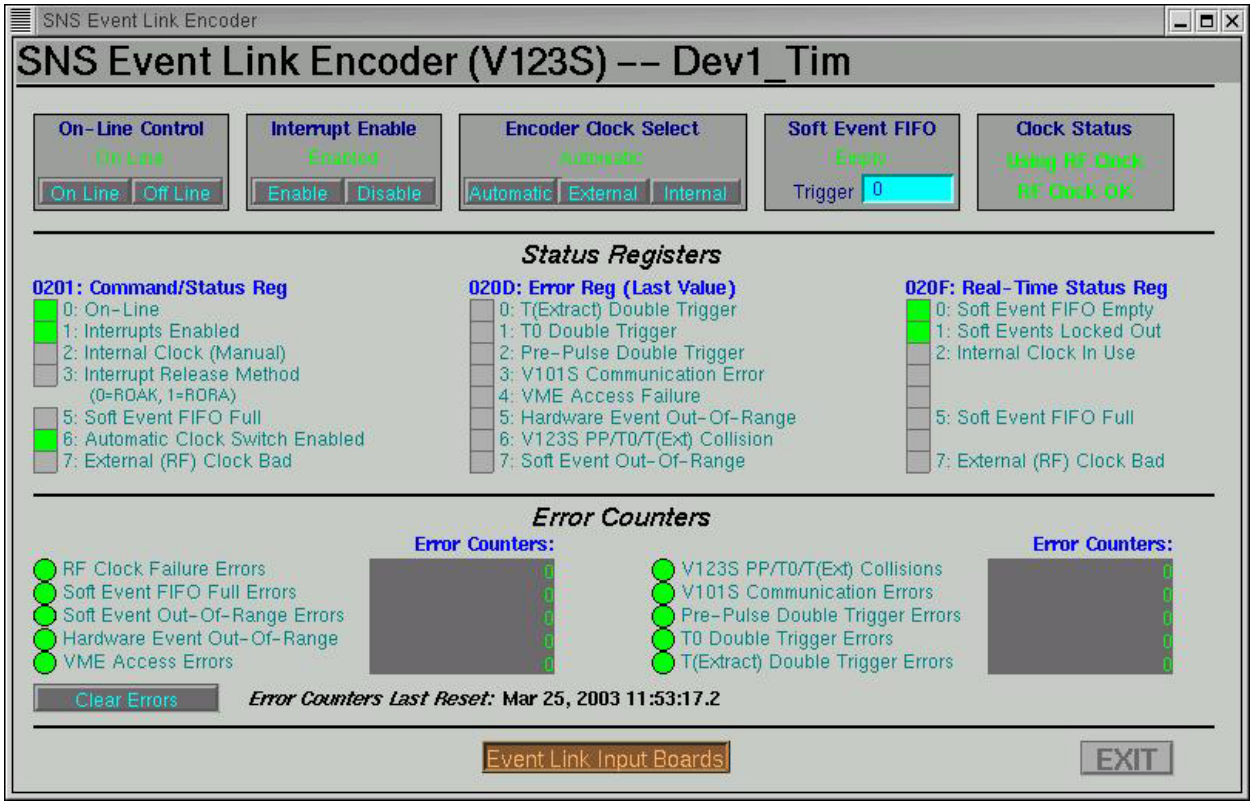


Figure 17
Event Encoder (V123S) Diagnostic Screen

For a correctly functioning V123S, the Command/Status register should have the following bits set:

- On Line
- Interrupt Enabled
- Automatic Clock Switch Enabled

The Error Register should be all zero, and the Real-Time Status Register should usually have "Soft Event FIFO Empty" set (although it may occasionally blink off). "Soft Events Locked Out" may either be set or not set, depending on when in the machine cycle the register read occurred.

Before the accumulation ring is built, there will be no external ring RF signal. During this period, the event link clock will be generated internally by the V123S module. When this occurs you may also see "Internal Clock In Use" and "External (RF) Clock Bad" indications. Although these indications display as error conditions, they can safely be ignored until we start running the ring.

The operator can generate "software events" (events numbered from 64 to 255) by writing the event number into the "Trigger" field of the "Soft Event FIFO" display.

The "Event Link Input Boards" button at the bottom of the display brings up a menu of the V101 boards. Selecting the first V101 board from the menu brings up the following screen:

Figure 18
Event Link Input Board (V101) Diagnostic Screen

The event name and the V101 channel are displayed on the left side of the screen. The actual event number transmitted on the event link is displayed next to the event name. A hardware event may be triggered manually by pushing its "Trigger" button on this screen – although the practice is discouraged during production.

The operator can also "Re-Map" the event number generated by a given V101 channel, although this practice is also highly discouraged unless you absolutely have to. One of the reasons it is discouraged is that the V123S module has a quirk about reading or writing the event map in the time between Cycle Start and Extract. This quirk could result in V123S bus timeouts, which could crash the IOC.

If you really, *really, REALLY,* ***REALLY*** need to change a channel's event number on-line, you must first enable event map writes by pressing the button in the lower left corner of the screen. This will activate the event number input fields and display a message about how changing event mappings could create event link errors. The event link driver software actually disables the event link just before it re-writes the map, and then re-enables the event link after the write (this is the other "safe" time to access the event map). When you have finished making your changes, disable event map writing again with the same button.

The "Lost Event" error counters displayed on the right side of the screen reflect how many times the V101 got a trigger on that channel but could not generate an event because it was already generating that event. These error counts could reflect noisy signals into the V101 input ports.

3.4 SOFTWARE

The EPICS driver and device support software for the SNS event link system reside in the files:

```
$IOCTOP/timingMaster/production/timingLib/src/drvEventLink.c
$IOCTOP/timingMaster/production/timingLib/src/devEventLink.c
```

These files are built into the "timingMasterLib" library which is loaded by the timing master IOC. The header file "drvEventLink.h" (in the same directory) contains the interface definitions and is installed in the directory:

```
$IOCTOP/timingMaster/production/include
```

The routines below are implemented in the drvEventLink.c module. Note that in some cases, the term "V101 Event Number" can refer to either a hardware or a software event.

status = eventLinkConfig (base, vector, level)

This routine is called from the vxWorks startup file to configure the event link system's VME address and interrupt vectors.

Arguments:

base = Base VME address for event link system.

vector = Base interrupt vector for event link system. Note: The vector number specified here should be the "base" vector for the event link system (which actually turns out to be the interrupt vector for the first V101 module and not the V123S). Because of the way the event link system allocates interrupt vectors, the vector number specified here needs to be divisible by 16. 0x20 is the vector reserved for the event link system in the timing master IOC (see section 2.1)

level = Interrupt request level.

Returns:

OK = Event link system successfully configured.

ERROR = Event link configuration had errors.

status = eventLinkInit ()

complete the event link system hardware initialization, including setting up the interrupt vectors and enabling interrupts. This routine is part of the drvEventLink EPICS driver entry table and is normally only called by the EPICS iocInit process.

Returns:

OK = Event link hardware successfully initialized.

ERROR = Error during event link hardware initialization.

status = eventLinkReport (level)

Reports on the module ID, serial number, address and configuration of each card in the event link system.. This routine is part of the drvEventLink EPICS driver entry table, and is invoked whenever the caller issues the "dbior" command. It may also be directly invoked from the vxWorks shell

Arguments:

level = (int) Level of detail to report. Not used.

Returns:

Always returns OK.

status = eventLinkGet (event, tag, &buffer)

This is the generic driver routine for returning scalar information from the event link driver and hardware.

Arguments:

event = The "V101 Event" number of the event that this function should apply to.
tag = Function code describing what to read (tag values are defined in drvEventLink.h).
buffer = Address of an unsigned byte array buffer to receive the data.

Returns:

Returns "OK" if the read succeeded.
Returns "ERROR" if the read failed.

status = eventLinkGetBytes (event, tag, size, &buffer)

This is the generic driver routine for returning byte array information from the event link driver and hardware.

Arguments:

event = The "V101 Event" number of the event that this function should apply to.
tag = Function code describing what to read (tag values are defined in drvEventLink.h).
size = Number of bytes to read.
buffer = Address of an unsigned byte array buffer to receive the data.

Returns:

Returns "OK" if the read succeeded.
Returns "ERROR" if the read failed.

status = eventLinkPut (event, tag, value)

This is the generic driver routine for writing scalar information to the event link driver and hardware.

Arguments:

event = The "V101 Event" number of the event that this function should apply to (tag values are defined in drvEventLink.h).
tag = Function code describing what to do.
value = Value (unsigned longword) to write.

Returns:

Returns "OK" if the write succeeded.
Returns "ERROR" if the write failed.

status = eventLinkPutBytes (event, tag, size, &buffer)

This is the generic driver routine for writing byte array information to the event link driver and hardware.

Arguments:

event = The "V101 Event" number of the event that this function should apply to.
tag = Function code describing what to write.
size = Number of bytes to write.
buffer = Address of an unsigned byte array buffer containing the data to write.

Returns:

Returns "OK" if the write succeeded.
Returns "ERROR" if the write failed.

veto = eventLinkArmErrors ()

This routine is called by the SNS timing master sequencer to re-arm event link error interrupts prior to the next machine cycle. The event link driver “throttles” event link errors to keep them from overwhelming the system and causing work queue panics. It does this by disabling interrupts whenever it gets an error interrupt. At the end of every machine cycle, the timing master sequencer calls this routine to re-arm event link error interrupts. This effectively throttles the event link error rate down to 60 Hz.

Returns:

veto = "Veto" mask if there were event link or RF clock synch errors during the last cycle. This veto mask is incorporated into the RTDL “Last Cycle Veto” frame (see section 4.1.9).

eventLinkClearErrors ()

Clears all event link error counters and flags.

eventLinkMapRefresh ()

This routine is called by the SNS timing master sequencer to refresh the in-core event mapping table from the hardware.

While the V123S card is on-line, you cannot read or write the event mapping table between Cycle Start and Extract without causing a bus error. In order to make the event map displayable, the timing master sequencer calls this function every 10 seconds to read the hardware event map and copy it into an in-core buffer where it can be accessed by the event link device support. Since the timing master sequencer is always called after the Extract event, it is safe for it to access the event map table.

4 REAL TIME DATA LINK

The SNS Real Time Data Link (RTDL) is modeled after the RHIC Real Time Data Link [2]. It is a 10 Mhz, bi-phase mark encoded, differential PECL, serial link.

An *RTDL Frame* consists of 41 bits. Its payload is a 24 bit data word representing an SNS accelerator parameter pertaining to the upcoming machine cycle. The rest of the RTDL frame consists of a start bit, an 8-bit frame number, and 8-bit frame CRC. The format of an RTDL frame is:

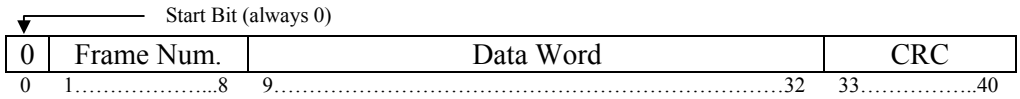


Figure 19
RTDL Frame Format

Before the start of each machine cycle, a series of frames about the upcoming cycle is transmitted on the RTDL. In the current SNS timing system, an RTDL transmission could contain up to 128 frames. 255 frames are possible if a third VME crate is added to the timing system. The last RTDL frame of the transmission is always frame 255 and contains a 24-bit "Message CRC" which is computed by the timing master IOC.

4.1 RTDL FRAME DEFINITIONS

These are the frames that currently constitute the RTDL transmission.

4.1.1 Time Stamp (Frames 1-3)

The first three RTDL frames contain the 64-bit EPICS timestamp to be applied at the next Cycle-Start event. The 64-bit timestamp is divided into two 32-bit long words as follows:

Seconds Past Epoch
Nanoseconds

Figure 20
EPICS Timestamp Format

The 64-bit EPICS timestamp is transmitted on three RTDL frames. The RTDL timestamp encoding is shown below:

RTDL Frame 1:	High-Order 24-bits of Seconds		
RTDL Frame 2:	MSB Nanoseconds	LSB Cycle Number	LSB Seconds
RTDL Frame 3:	Low-Order 24-bits of Nanoseconds		

Figure 21
RTDL Timestamp Format

The timing master IOC inserts the least significant byte of the current machine cycle number into the middle byte of the 2nd RTDL timestamp frame. This guarantees that each timestamp will be unique, even if the timing master IOC does not have an external clock source and is relying on the vxWorks clock for its system time (the vxWorks clock normally runs at 60 Hz and could return the same data for two consecutive cycles). This is important to the MPS system, which will fault if it detects that the RTDL timestamp is not changing (which it interprets to mean that the timing master IOC is sick). Anyone else decoding the RTDL timestamp should disregard the middle byte of RTDL frame 2.

4.1.2 Ring Revolution Period (Frame 4)

RTDL frame 4 contains the ring revolution period in picoseconds. This value is useful for converting timing system units into real time. The ring revolution period is acquired my measuring the frequency of the event link clock. This process is described in section 7.

4.1.3 MPS Mode (Frame 5)

The MPS Mode number is broadcast on RTDL frame 5. This frame is primarily of interest to the MPS system. MPS uses the value in this frame to set the masks of which conditions to ignore and which conditions are relevant for the current operating mode.

The MPS system wants to make sure that the mode information coming from the RTDL is correct and does not contain any transmission errors. For this reason, the mode information is encoded three ways in RTDL frame 5. The low order byte contains the unadulterated mode number. The middle byte contains the mode times two plus one. The high order byte contains the "ones compliment" of the mode. This encoding is show below in Figure 22:

One's Compliment of Mode	(Mode × 2) + 1	Mode

Figure 22
RTDL Mode Frame Format

4.1.4 60 Hz. Phase Error (Frame 6)

This frame represents the distance in nanoseconds between the actual 60 Hz line frequency zero crossing and the "smoothed" zero crossing signal from the master reference generator (see section 5). This value is read from the master reference generator every cycle. It can be used to veto pulses or raise MPS conditions if the phase error gets outside the acceptable tolerance.

4.1.5 Beam Width (Frame 7)

This frame represents the width in turns of the current beam flavor. It is mainly used to synchronize diagnostic gates. If beam is scheduled to be on in the next cycle, this frame reflects the width of the particular flavor of beam scheduled for that cycle. If beam is off, this frame reflects the width of the primary flavor.

4.1.6 IOC Reset (Frame 15)

Each SNS Utility Module has a 24-bit "IOC Reset" address associated with it. This address is set by jumpers on the utility module board. When the utility module firmware detects a value in RTDL frame 15 that matches its own IOC Reset address, it will assert the VME SYSRESET line, causing the IOC to reboot. This feature allows an IOC to be rebooted remotely, even if its software is completely hung up. It will not, of course, allow for the remote reboot of a powered-off IOC.

IOC reboots are requested by writing the reset address of the specified IOC to a special record maintained by the RTDL encoder (V105S) driver. The record writes the IOC reset address to a vxWorks *Message Queue*, which was also created by the RTDL encoder driver. Each cycle, the timing master sequencer reads this message queue, and if the queue is not empty, the next value is placed in RTDL frame 15.

RTDL frame 15 is broadcast every cycle (as are all the RTDL frames defined here). Normally its value will be 0, indicating that no IOCs should be rebooted.

4.1.7 Pulse Flavor (Frame 17)

RTDL frame 17 contains the "pulse flavor" of the next cycle. Pulse flavors are described in section 1.1.4.

4.1.8 RF Gate Widths (Frames 18-21)

The widths of the primary RF gates are broadcast in RTDL frames 18 through 21. These values can be used by the low-level and high-power RF IOCs to synchronize their local timing gates with the gate width values set in the timing master EDM screen. The assignments for the four RF gate width frames are shown below:

RTDL Frame 18:	Warm Linac High-Power RF Gate Width (turns)
RTDL Frame 19:	Warm Linac Low-Level RF Gate Width (turns)
RDTL Frame 20:	Cold Linac High-Power RF Gate Width (turns)
RTDL Frame 21:	Cold Linac Low-Level RF Gate Width (turns)

Figure 23
RF Gate Width Frames

4.1.9 Last Cycle Veto (Frame 24)

RTDL Frame 24 describes possible veto conditions to the experimental stations. Unlike the other RTDL frames, this frame pertains to the *previous* cycle rather then the *next* cycle. Each bit in this frame describes a reason why one or more experimental stations may want to veto data from the previous cycle. These bits are defined in the **snsTiming.h** file. 12 veto bits are currently defined. Their values and symbolic names are listed below:

Bit	Name	Description
0	VETO_NO_BEAM	No beam was delivered on the previous pulse.
1	VETO_NOT_TARGET_1	Beam was delivered to target 2 (not to target 1)
2	VETO_NOT_TARGET_2	Beam was delivered to target 1 (not to target 2)
3	VETO_DIAGNOSTIC_PULSE	Beam was a "reduced intensity" diagnostic pulse
4	VETO_PHYSICS_PULSE_1	Beam was one of the special physics study pulses
5	VETO_PHYSICS_PULSE_2	Beam was one of the special physics study pulses (the other type)
6	VETO_MPS_AUTO_RESET	Beam was interrupted by an "Auto Reset" MPS trip (fast protect)
7	VETO_MPS_FAULT	Beam was interrupted or not delivered because of a "Latched" MPS trip
8	VETO_EVENT_LINK_ERROR	Timing system detected corruption on the event link
9	VETO_RING_RF_SYNCH	Timing system has lost synch with the Ring RF signal
10	VETO_RING_RF_FREQ	Measured ring RF frequency is outside acceptable range
11	VETO_60_HZ_ERROR	60 Hz line phase error is out of tolerance

Figure 24
Veto Bit Definitions

4.1.10 Cycle Number (Frame 25)

The cycle number within the *Super Cycle* is broadcast on RTDL frame 25. An SNS super cycle is 600 cycles long. Therefore, this number will always be between 0 and 255. This information is useful for client IOCs that implement local rep-rates.

4.1.11 Message CRC (Frame 255)

The last RTDL frame transmitted is always frame 255. This frame contains a 24-bit CRC of the entire RDTL transmission. The CRC value is computed by the timing master. Details of the CRC computation can be found in [7].

4.2 HARDWARE

The RTDL hardware consists of one *RDTL Encoder* module (V105S) and up to 16 *RTDL Input* modules (V206). The V206 modules contain the data to be sent on the RTDL. The V105S module formats this data into RTDL frames and transmits it upon receipt of an external trigger signal.

The V105S module resides in slot 4 of the upper timing master crate. The V206 modules occupy slots 5 through 20. The V105S and V206 modules communicate over a P2 backplane that spans the upper portion of the VME P2 connectors between slots 4 and 21. Slot 21 is reserved for another VME bridge module, in case we need more than 128 RTDL frames.

Each V206 module is capable of generating 8 RTDL frames. Consequently, if all 16 V206 modules are in place, the RTDL transmission can be as long as 128 frames (136 frames if you install another V206 in slot 21). It is possible to generate up to 255 RTDL frames if another crate is bridged to the timing master IOC.

The RTDL P2 backplane installation is shown below in Figure 25:

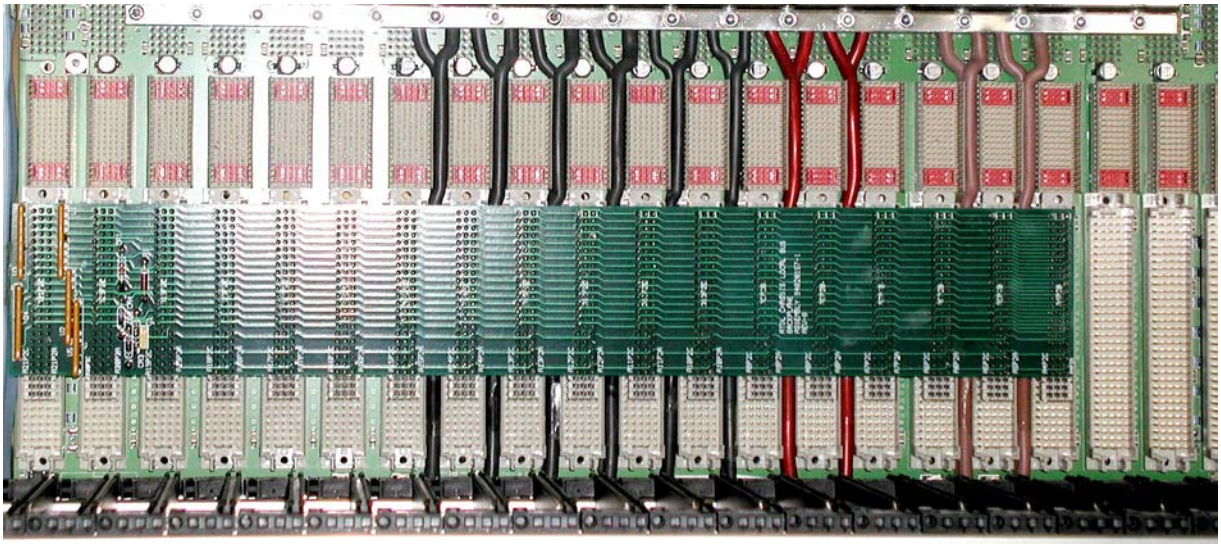


Figure 25
RTDL P2 Backplane Installed In Slots 4-21.

4.2.1 RTDL Encoder Module (V105S)

The V105S module is responsible for generating the RTDL carrier signal. This is a 10 Mhz. differential PECL, bi-phase mark encoded, serial link. Unlike the event link, the RTDL clock is not derived from the ring RF. It takes about 5 microseconds to transmit one RTDL frame. When computing the time between RTDL Xmit and RTLD Valid, the timing master allows 6 turns per frame.

The V105S contains an internal list of frame numbers. This list is built during system initialization and determines which frames are to be transmitted in and their order. The list is “zero-terminated”, so the maximum number of RTDL frames that can be sent at a time is 255.

When the V105S receives a TTL trigger on its “EXT TRIG” input, it cycles through its frame list. Using the P2 backplane connector, it requests data for each frame in the list. The V206 module

containing the requested frame will then respond with the frame's data word. The data is formatted into an RTDL frame with an 8-bit CRC appended, and sent out on the RTDL.

4.2.1.1 Connections

There are two connections on the front panel of the V105S module. A twinaxial BNC connector at the top provides the RTDL output and connects the module to the RTDL fanout chassis. A LEMO connector below it is the input port for the TTL trigger that controls when the RTDL transmission occurs. This trigger comes from a V124S gate (see section **Error! Reference source not found.**) and is generated by the "RTDL Xmit" event.

4.2.1.2 Jumper Settings

The VME A16 base address is set via seven jumpers located toward the back of the card (see Figure 26 below). The jumpers set base address lines A9 - A15 (512 byte boundary). With the board standing upright (as shown below), the high-order bit is at the top and the low order bit is toward the bottom. If a jumper is present, it represents a logical "0". If the jumper is removed, it represents a logical "1".

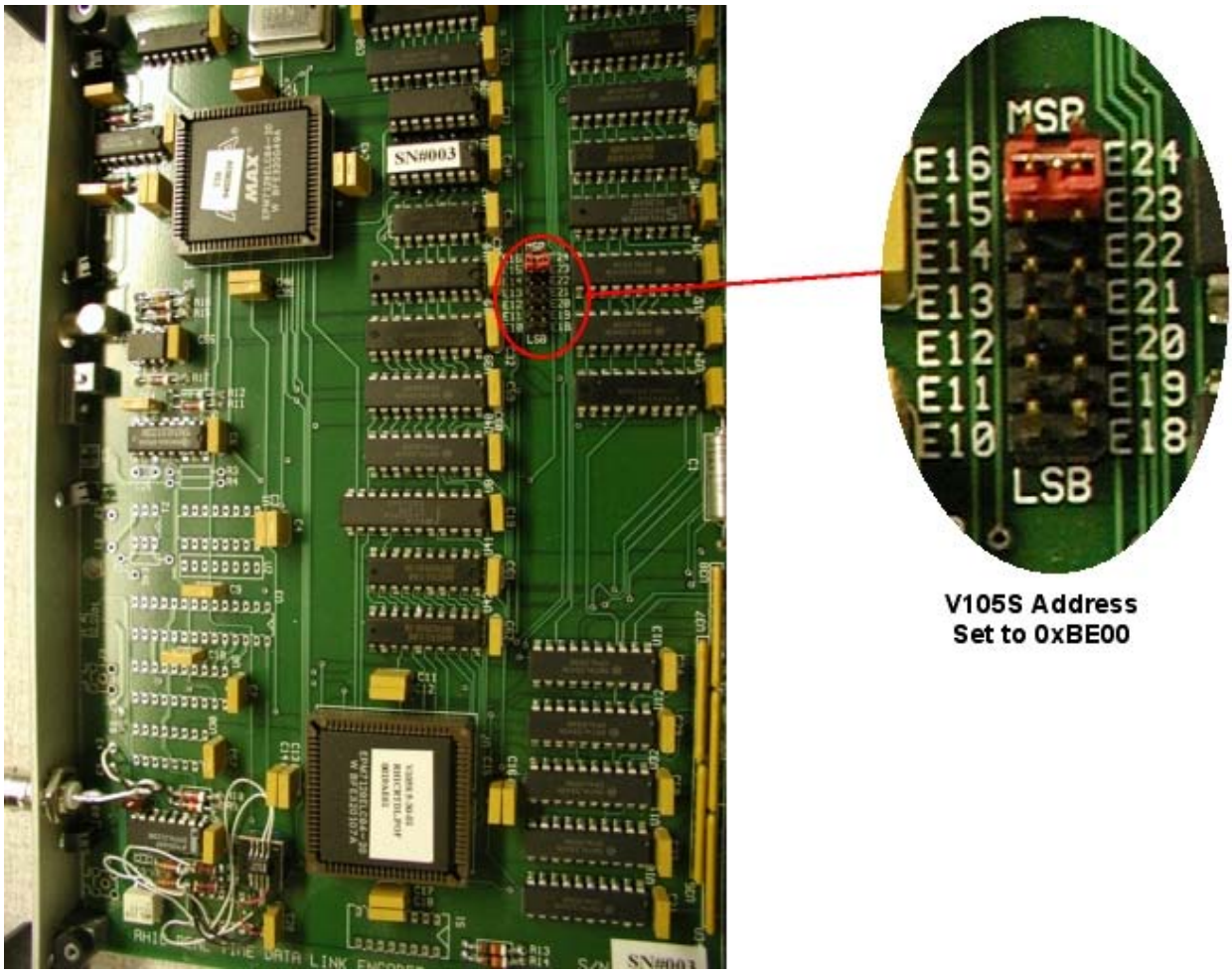


Figure 26
V105S Address Selection

4.2.2 RTDL Input Module (V206)

The V206 RTDL Input Module is the exact same module used at RHIC. No modifications have been made for the SNS.

4.2.2.1 Connections

Normally, there will be no external connections to the V206 modules. The V206 modules communicate with the V105S via a dedicated P2 backplane. A V206 module has eight Twinax BNC connectors on its front panel. Their primary purpose (apart from making the module hard to insert) is to allow automatic loading of the 8 RTDL frames via hardware function generators. This feature is not used at SNS.

4.2.2.2 Jumper Settings

The VME A16 base address is set via seven jumpers located toward the back of the card (see Figure 27 below). The jumpers set base address lines A9 - A15 (512 byte boundary). With the board standing upright (as shown below), the high-order bit is at the bottom and the low order bit is toward the top (this is just the opposite of the V105S board). If a jumper is present, it represents a logical “0”. If the jumper is removed, it represents a logical “1”.

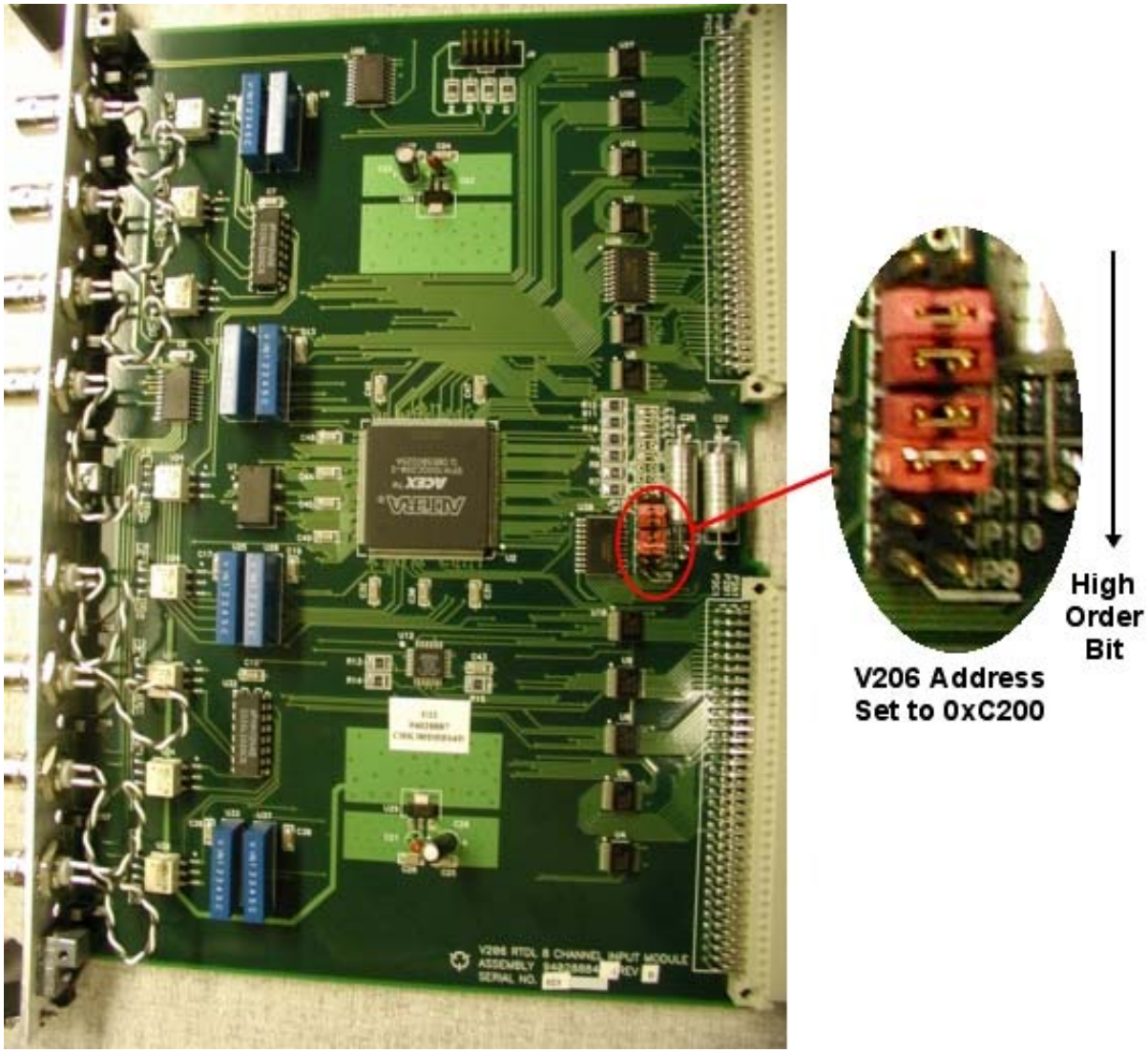


Figure 27
V206 Address Selection

4.3 DIAGNOSTIC SCREENS

Only diagnostic "expert" screens are available for the RTDL system. The diagnostic screens can be obtained by starting at the Timing Master EDM screen and selecting the "Timing System Diagnostics" related display button. From the Timing System Diagnostic screen, select the "RTDL Diagnostic Screen" related display button. This will bring up the RTDL Encoder (V105S) screen shown below:

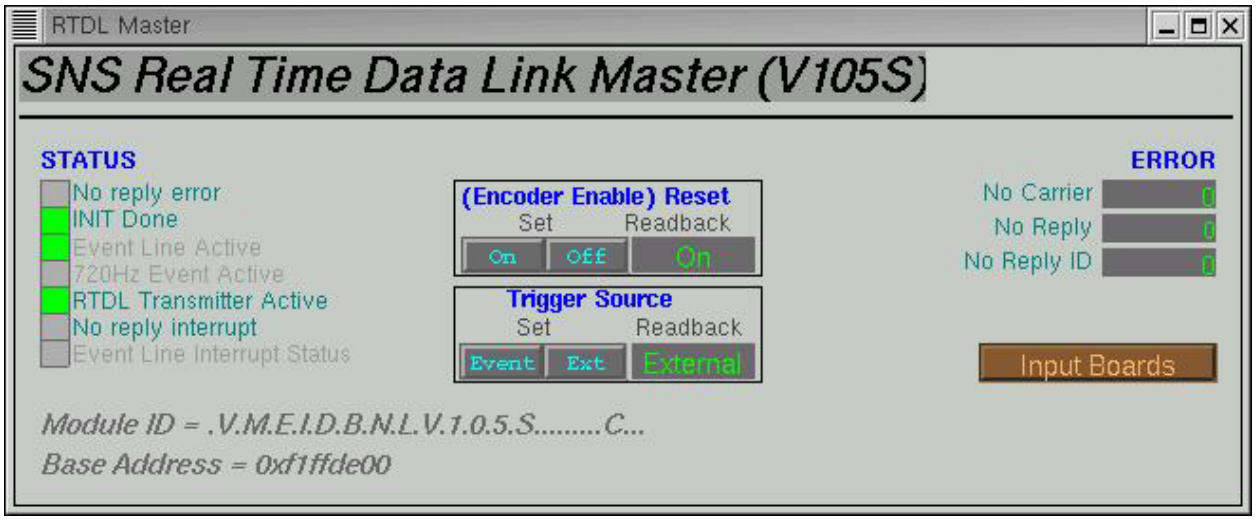


Figure 28
RDTL Encoder (V105S) Diagnostic Screen

On a correctly functioning V105S, the Encoder Enable readback should be "On" and the Trigger Source readback should be "External". The status bits for "INIT Done" and "RTDL Transmitter Active" should be on. A "No Reply" error indicates that one of the V206 input modules is not responding with a requested RTDL frame. The frame number of the latest "non-responding" frame is displayed in the "No Reply ID" field on the right hand side of the screen.

To see the individual RTDL inputs, select the "Input Boards" menu button on the lower right of the screen and chose the desired V206 card you wish to view. This will bring up the following screen:

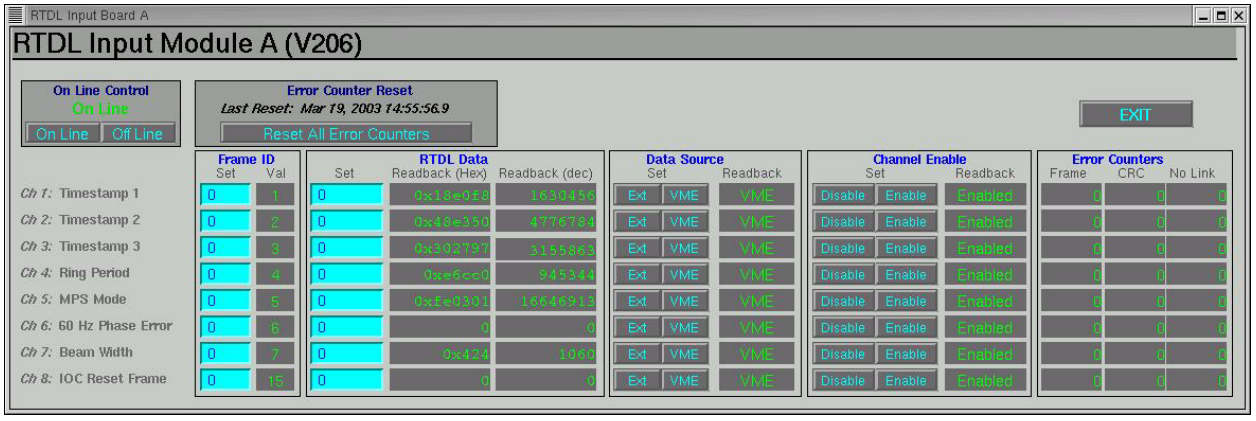


Figure 29
RDTL Input (V206) Diagnostic Screen

The channel number and frame description for each of the eight input channels are displayed on the right. The frame ID is displayed (and can be set) in the next box. The RTDL data words are displayed in both hexadecimal and decimal formats because some frames are better read in decimal and other frames are better read in hex. The frame data can be written for each channel. However, if the timing master is running, this data will be overwritten at the start of each new cycle. The "Data Source" field should always be "VME" and the "Channel Enable" field should always be "Enabled". The error counters only pertain to errors on the external input ports, so they should always read 0.

4.4 SOFTWARE

The EPICS software for the RTDL interface resides in the directory:

```
$IOCTOP/timingMaster/production/timingLib/src/
```

It is built into the "timingMasterLib" library which is loaded by the timing master IOC. The header file "drvRTDL.h" (in the same directory) contains the interface definitions and is installed in the directory:

```
$IOCTOP/timingMaster/production/include
```

4.4.1 RTDL Encoder (V105S)

The epics driver and device support for the RTDL encoder are in the files

```
$IOCTOP/timingMaster/production/timingLib/src/drvV105.h
$IOCTOP/timingMaster/production/timingLib/src/devV105.c
$IOCTOP/timingMaster/production/timingLib/src/devV105.h
```

The V105 device driver requires one call from the startup command script to initialize the card.

status = V105DevCreate(card, base , vector, level)

This routine is called from the vxWorks startup file to configure the V105 base address, interrupt vector, and interrupt level .

Arguments:

card	=	Card number (as specified in the EPICS record INP/OUT link fields).
base	=	Base address in A16 space.
vector	=	Interrupt vector.
level	=	Interrupt request level.

Returns:

OK	=	V105 card successfully configured.
ERROR	=	V105 card not installed or functioning.

In addition to the initialization function there are two functions implemented to support Epics driver support. The Epics dbd file statement "driver (drvV105)" binds the driver entry point table into IOC cores device list. Two functions, drvV105Report() and drvV105Init(), implement the device support layer. Record support for bi, bo, longin, and longout record types is implemented in devV105.c.

4.4.2 RTDL Input (V206)

The EPICS driver and device support software for the V206 module reside in the files:

```
$IOCTOP/timingMaster/production/timingLib/src/drvV206.c
$IOCTOP/timingMaster/production/timingLib/src/devV206.c
```

The following routines are implemented in the drvV206.c module:

status = v206Config (card, base, vector, level)

This routine is called from the vxWorks startup file to configure the V206 base address, interrupt vector, and interrupt level .

Arguments:

card = Card number (as specified in the EPICS record INP/OUT link fields).
base = Base address in A16 space.
vector = Interrupt vector.
level = Interrupt request level.

Returns:

OK = V206 card successfully configured.
ERROR = V206 card not installed or functioning.

status = v206Init ()

Initialize the V206 hardware, including setting up the interrupt vectors and enabling interrupts. This routine is part of the drvV206 EPICS driver entry table and is normally only called by the EPICS iocInit process.

Returns:

OK = V206 card successfully initialized.
ERROR = V206 card not installed or functioning.

status = v206Report (level)

Reports on the V206 card's ID, serial number, address and configuration. This routine is part of the drvV206 EPICS driver entry table, and is invoked whenever the caller issues the "dbior" command. It may also be directly invoked from the vxWorks shell

Arguments:

level = (int) Level of detail to report. Not used.

Returns:

Always returns OK.

V206_HANDLE = v206Handle (card)

Returns the driver handle for the specified V206 card.
Returns NULL if the specified card was not configured (via the v206Config function).

Arguments:

card = Card number to return the handle for.

status = v206Get (handle, channel, tag, &value);

This is the generic driver routine for returning scalar information from the V206 driver.

Arguments:

handle = Driver handle to the V206 card.
channel = Channel number (1-8) of the input channel we want to get information about. Channel 0 is used to return card-specific (as opposed to channel-specific) information.
tag = Function code describing what information we want (tag values are defined in drvRTDL.h).
value = Address of an unsigned longword to receive the information.

Returns:

Returns "OK" if the read succeeded.
Returns "ERROR" if the read failed.

status = v206GetBytes (handle, channel, tag, size, &buffer)

This is the generic driver routine for returning byte array information from the V206 driver.

Arguments:

handle = Driver handle to the V206 card.
channel = Channel number (1-8) of the input channel we want to get information about. Channel 0 is used to return card-specific (as opposed to channel-specific) information.
tag = Function code describing what information we want (tag values are defined in drvRTDL.h).
size = Size of the buffer for accepting the data (in bytes).
buffer = Address of buffer to receive the information.

Returns:

Returns "OK" if the read succeeded.
Returns "ERROR" if the read failed.

status = v206Put (handle, channel, tag, value)

This is the generic driver routine for writing information to the V206 card.

Arguments:

handle = Driver handle to the V206 card.
channel = Channel number (1-8) of the input channel we want to write to. Channel 0 is used to write card-specific (as opposed to channel-specific) information.
tag = Function code describing what we want to write (tag values are defined in drvRTDL.h).
value = Unsigned longword containing the data to write.

Returns:

Returns "OK" if the write succeeded.
Returns "ERROR" if the write failed.

5 MASTER REFERENCE GENERATOR

The master reference generator has not been completely implemented at the time of this writing. Consequently, this section only contains the design specifications.

The purpose of the Master Reference Generator is to monitor the AC line frequency and produce a stable reference pulse that does not change faster than the T0 neutron choppers can keep up with, and that does not stray too far from the actual zero crossing. The current operating requirements are that the reference pulse should not change any faster than 1 millihertz per second, and that it not stray more than ±500 microseconds from the actual zero crossing. The precise requirements will not be known until we get some operating experience with the klystrons and the neutron choppers, so it is important that these limits be adjustable.

The output of the master reference generator is a TTL signal that will drive the "Pre-Pulse" input on the V123S event link encoder module (this will later become the "Cycle-Start" input on future versions of the V123S). Before we get to the V123S, however, the master reference generator applies two additional delays to the reference pulse. The first is a *Reference Phase Delay*. This is a more or less constant delay that allows us to adjust the phase of the reference pulse relative to the AC line phase. The second delay is the *Ring RF Compensation Delay*. This delay is intended to compensate for the fact that the master reference generator's output pulse controls the start of the machine cycle, but it is actually the end of the machine cycle (the Extract event) that we really want to keep stable. Changes in the ring RF frequency will cause the machine cycle to increase or decrease in actual time, even though the "turn time" remains at constant 5,050. The difference between the ring revolution time at the highest and lowest design frequencies is about 62.2 nanoseconds per turn. Since there are 5,050 turns between Cycle-Start and Extract, the variation in the machine cycle length can be up to 314 microseconds.

Each cycle the timing master will measure the frequency of its clock signal (the 32 X Frev signal input to the V123S module). A compensating delay value will be set in the "Ring RF Compensation Delay" register of the master reference generator. This value will increase the delay between zero crossing and Cycle-Start if the ring RF frequency increases, and decrease the delay between zero crossing and Cycle-Start if the ring RF frequency decreases.

The relationships between the smoothed zero crossing signal, the reference phase delay, the ring RF compensation delay, and the actual output signal are illustrated in the figure below:

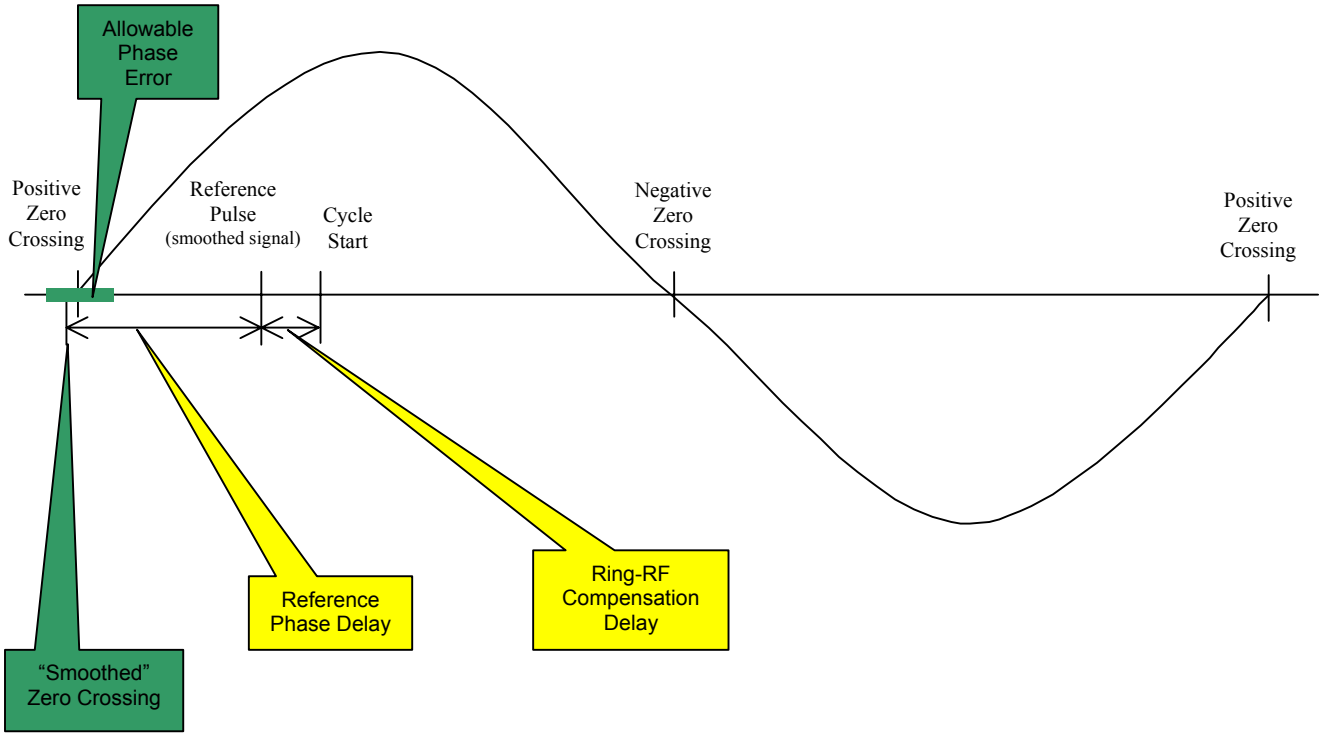


Figure 30
Master Reference Generator Time Line

5.1 OUTPUT SIGNALS

The master reference generator produces at least four TTL output signals. These are:

- **Raw Positive Zero Crossing:** This is mostly for diagnostic purposes. It can be used to check the zero-crossing circuitry and to do statistical measurements of the site power.
- **Reference Pulse:** This is the smoothed reference pulse, after the reference phase delay is applied but before the Ring-RF compensation delay. This is also a diagnostic output and can be used to check the stability of the master reference generator.
- **Cycle-Start Pulse:** This is the signal that goes to the Pre-Pulse/Cycle-Start input on the V123S event link encoder module. It is the smoothed zero-crossing pulse after the reference phase and ring-RF compensation delays are applied.
- **Error Signal:** A TTL signal that is high whenever the master reference generator detects that it has lost phase lock with the 60 Hz AC line, or when the allowable phase error has been exceeded.

For 120 Hz operation, an additional three outputs, corresponding to the first three outputs above, will be needed for the negative zero crossing.

5.2 VME REGISTERS

The A16 address space between 0x2800 and 0x2FFF in the timing master IOC has been reserved for the master reference generator. This space will contain the VME registers used to communicate between the master reference generator and the timing master IOC. All registers are 16 bits wide and addressed via VME A16/D16 read/write cycles. The following VME registers are required:

Status	Read Only	Bitmask	Bitmask describing the status of the master reference generator. Contains bits for error conditions such as lost phase lock, phase error out of tolerance, or anything else that might indicate the module isn't performing as expected.
Phase Error	Read Only	Bipolar	The distance between the actual zero crossing and the smoothed zero crossing pulse. One bit represents 25 nanoseconds. This value (converted to nanoseconds) is transmitted on RTDL frame 6 by the timing master IOC.
Phase Limit	Read/Write	Unipolar	The allowed distance between the actual zero crossing and the smoothed zero crossing pulse. One bit represents 25 nanoseconds. If the actual phase error exceeds this limit, the master reference generator will set an error bit in the "status" register and assert the "Error" output signal until the phase error returns to the specified tolerance.
Max Slew Rate	Read/Write	Unipolar	The maximum rate the smoothed zero crossing pulse will be allowed to slew as it pursues the real zero crossing. The units for this register are somewhat unclear. The spec is in millihertz per second, but the current prototype uses a "sigma" number from 0 to 99. The actual units may need to be determined by the implementation.
Reference Phase Delay	Read/Write	Unipolar	The fixed phase delay between the smoothed zero crossing pulse and the reference pulse output of the master reference generator. One bit represents on microsecond.
Ring RF Compensation Delay	Read/Write	Unipolar	The delay between the reference pulse output and the "Cycle-Start" output of the master reference generator. This value is written once per cycle by the timing master IOC based on its measurement of the ring RF clock rate. One bit represents 25 nanoseconds.

Master Reference Generator Registers

5.3 EXTERNAL INTERFACES

The only external interface required is an interface to the zero crossing transformer. This transformer resides in a 1U chassis mounted in the timing master IOC cabinet.

6 GLOBAL POSITIONING SYSTEM INTERFACE

The timing master IOC uses a global positioning system (GPS) to get the time of day that it broadcasts on the RTDL. The GPS receiver also serves as the site NTP server.

6.1 HARDWARE

The GPS receiver is a TrueTime XL-DC Time and Frequency Receiver. This is a 2U chassis that is mounted in the timing master rack. The interface to the timing master IOC is a TrueTime VME-SG2 VME card. The card interfaces to the receiver via the receiver's external IRIG-B output. The VME-SG2 card can also function as a stand-alone time source.

The VME-SG2 card has two sets of "Freeze" registers that can be used to capture and read the current time. The first set of "Freeze" registers can be triggered via a VME read operation. This set can be used by EPICS device-support routines to capture and display the current time in a string record. The second set of "Freeze" registers is triggered by an external TTL signal. A timing gate synchronized with the "Cycle-Start" event is used to trigger these registers. The captured value is then used to compute the time of the next "Cycle-Start" event and distribute this time on the "Real Time Data Link" (RTDL).

6.1.1 Connections

The GPS antenna and the IRIG-B signal to the VME-SG2 module connect to two BNC connectors in the rear panel of the GPS receiver module. These connections are show below in Figure 31.

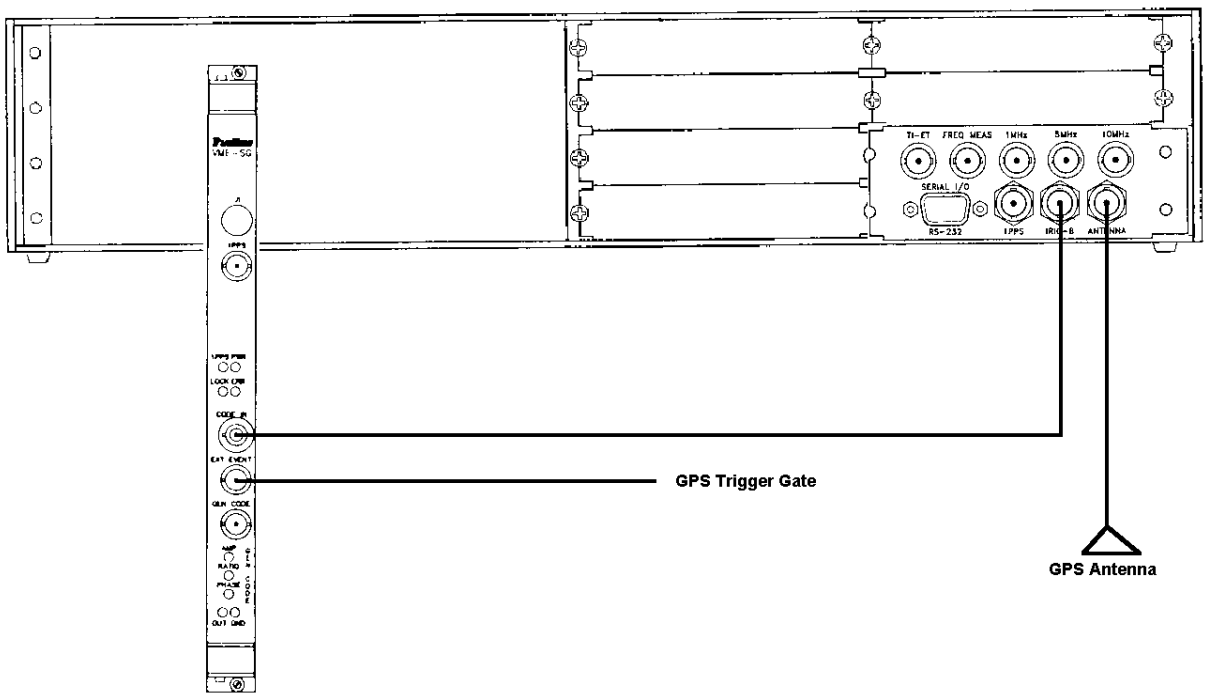


Figure 31
Connections to GPS Receiver and VME-SG2

The GPS antenna connects to the "ANTENNA" BNC port on the back of the GPS receiver. The "CODE-IN" port on the front panel of the VME-SG2 connects to the "IRIG-B" BNC port on the back of the GPS receiver.

The “EXT EVENT” port on the VME-SG2 connects to the “GPS Trigger” gate. The “GPS Trigger” gate occurs at the same time as the gate that triggers the Cycle-Start event. When this gate fires, the current GPS time is copied into an internal register in the VME-SG2. The timing master IOC reads this value, adds 16.667 milliseconds to it, and broadcasts it on the RTDL as the timestamp for the next Cycle-Start event.

6.1.2 Jumper Settings

The VME-SG2 A16 base address is set by a dip-switch located at the rear of the card. The switches set base address lines A8 - A15 (256 byte boundary). With the board standing upright (as shown below), the high-order bit (Switch 8) is on the bottom and the low order bit (Switch 1) is on the top. A logical “0” is represented by moving the switch to the right (ON). A logical “1” is represented by moving the switch to the left (OFF).

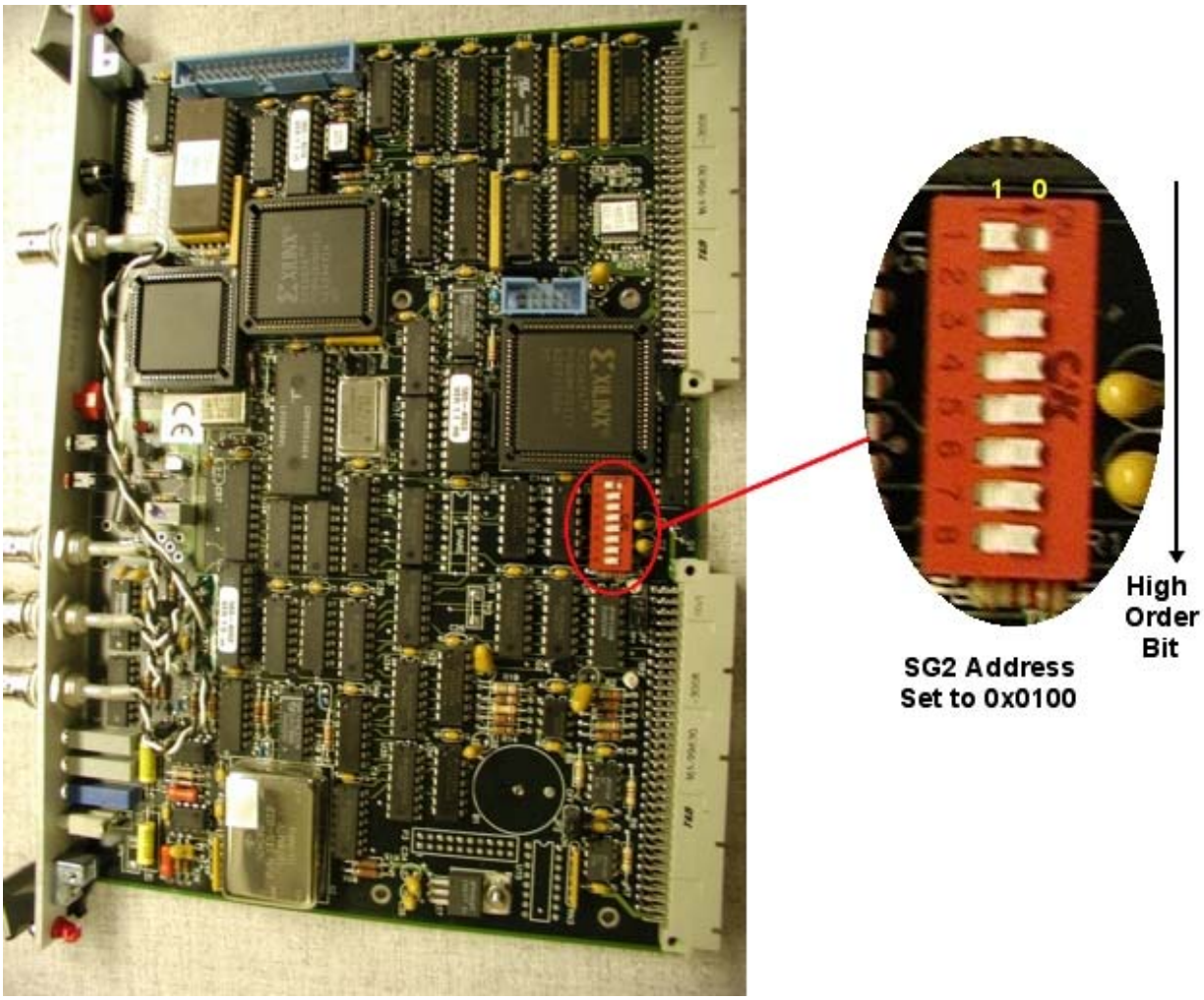


Figure 32
VME-SG2 Address Selection

6.1.3 Configuring the NTP Server

T.B.S.

6.2 SOFTWARE

The EPICS driver for the GPS interface resides in the directory:
`$IOCTOP/timingMaster/production/timingLib/src/drvGps.c`

It is built into the "timingMasterLib" library which is loaded by the timing master IOC. The header file "drvGps.h" (in the same directory) contains the interface definitions and is installed in the directory:
`$IOCTOP/timingMaster/production/include`

The following routines are implemented in the drvGps.c module:

gpsConfigure (base, gpsFlag);

This routine is called in the timing master IOC's startup file to configure the VME-SG2 card.

Arguments:

base = Base address (in A16 space) of the VME-SG2 registers. The base address must be a multiple of 256 bytes.

gpsFlag = If TRUE, then get time from the connected GPS receiver. If FALSE, then generate the time from the internal 10 Mhz clock. This value should be FALSE (0) if you do not have a GPS receiver unit connected to the VME SG2 card.

status = gpsInit ();

Initialize the VME-SG2 hardware. This routine is part of the drvGps EPICS driver entry table and is normally only called by the EPICS iocInit process.

Returns:

OK = VME-SG2 card successfully initialized.

ERROR = VME-SG2 card not installed or functioning.

status = gpsReport (level);

Reports on the VME-SG2 card's address and configuration. This routine is part of the drvGps EPICS driver entry table, and is invoked whenever the caller issues the "dbior" command. It may also be directly invoked from the vxWorks shell

Arguments:

level = (int) Level of detail to report. Not used.

Returns:

Always returns OK.

status = gpsGetEventTime (&eventTime);

Retrieves the time (as a 64-bit EPICS timestamp) of the last "Cycle-Start" Event. This routine assumes that the "EXT EVENT" input on the VME-SG2 card is connected to a timing gate triggered by "Cycle-Start".

Arguments:

eventTime = (struct timespec*) 64-bit EPICS timestamp corresponding to the time of the last "Cycle-Start" Event.

Returns:

OK = Cycle-Start time retrieved from the VME-SG2 card.

ERROR = VME-SG2 card not installed or functioning.

value = gpsGetRegister (offset);

Retrieves a value from the VME-SG2 register at the specified offset, or 0xffff if there is no VME-SG2 card configured.

Arguments:

offset = Byte offset of the desired VME-SG2 register. These offset values are defined in the drvGps.h header file.

gpsPutRegster (offset, value);

Writes the specified value into the VME-SG2 register at the specified offset.

Arguments:

offset = Byte offset of the desired VME-SG2 register. These offset values are defined in the drvGps.h header file.

value = (uint16_t) value to write to the register.

7 RING REVOLUTION PERIOD MEASUREMENT

The timing master broadcasts the ring revolution period on the RTDL, where it can be used by the client IOCs to translate timing system units into microseconds. The timing master also uses this value to compute the ring RF compensation delay that it writes to the master reference generator (see section 0). We desire to know the ring revolution period to the nearest picosecond.

7.1 HARDWARE

The hardware used to measure the ring revolution period is the Joerger VS64 counter/scalar module. There are two methods we can use to measure the ring revolution period with this module. The first method requires two V124S gates that fire exactly 16,000 turns apart. The first gate starts a counter on channel one which counts the module's internal 50 Mhz clock for 16 milliseconds. The second gate causes the counter to "transfer" its current count into an output register exactly 16,000 turns after the trigger gate. The ring revolution period in picoseconds is obtained by multiplying the resulting count by 1.25 (20,000 picoseconds per clock / 16,000 turns). This method is accurate to ±1 picosecond, which is good enough for our current needs. This is the method we currently use.

If greater accuracy is required, we could take the "recovered clock" output signal from the front panel of a V124S module, convert it from differential PECL to TTL, and count that signal for 31.25 milliseconds. The resulting count will be the ring revolution frequency in Hertz, which can be easily converted to picoseconds with a full 6 digits of precision. The disadvantage is that we can now only update the revolution period every other cycle.

It should be noted that we can not use the first method (the "two gate" method) to integrate any longer than 16.67 milliseconds because the Cycle-Start event re-synchs the revolution counter, which could shorten the length of one of the measured turns. It should also be noted that the "two gate" method will still work if we run the timing system at 120 Hz, since the "Alternate Cycle-Start" event does not reset the revolution counter.

7.1.1 Connections

The connections for the "Two Gate" method are shown below. Two V124S gates are required. The first gate is the "Counter Trigger" gate. Its job is to reset the channel 1 counter and then trigger it for a maximum integration time of 16.166 milliseconds. It connects to the "Reset Input" port of the VS64 and is daisy-chained to the "Gate/Trigger" port. This gate is triggered by the Cycle-Start event and has a revolution delay of 2, a sub-revolution delay of 1, and a fine delay of 0. Its width is set to 1 sub-revolution and its polarity is set to "inverted" (low-true).

The second gate is the "Counter Transfer" gate. Its job is to read the counter exactly 16,000 turns after it was triggered. This gate connects to the "Xfer Input" port of the VS64. This gate is triggered by the Cycle-Start event and has a revolution delay of 16002 and a sub-revolution delay of 1. Its width is set to 1 sub-revolution and its polarity is set to "inverted" (low-true). The fine delay on this gate is used to calibrate the measurement. Its use is described below in section 7.1.3.

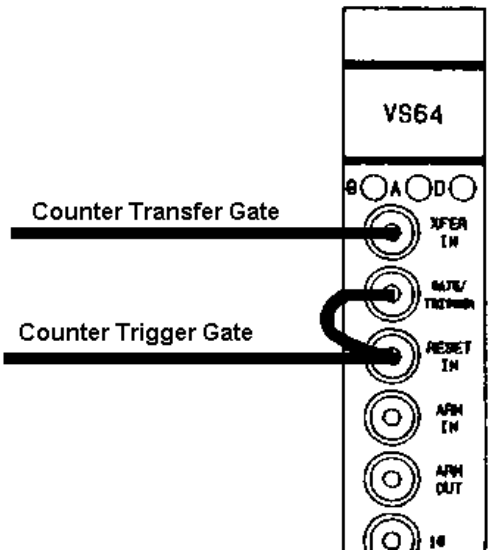


Figure 33
VS64 Connections For The "Two Gate" Measurement Method

7.1.2 Jumper Settings

The A16 base address is set with five jumpers at the back of the board (see the picture below). The jumpers select A11-A15 (2K boundary). With the board standing upright (as shown below), the high-order bit (A15) is on the bottom and the low order bit is on the top. These are three-post jumpers. A logical "1" is represented by jumpering the left-most post to the middle post. A logical "0" is represented by jumpering the right-most post to the middle post.

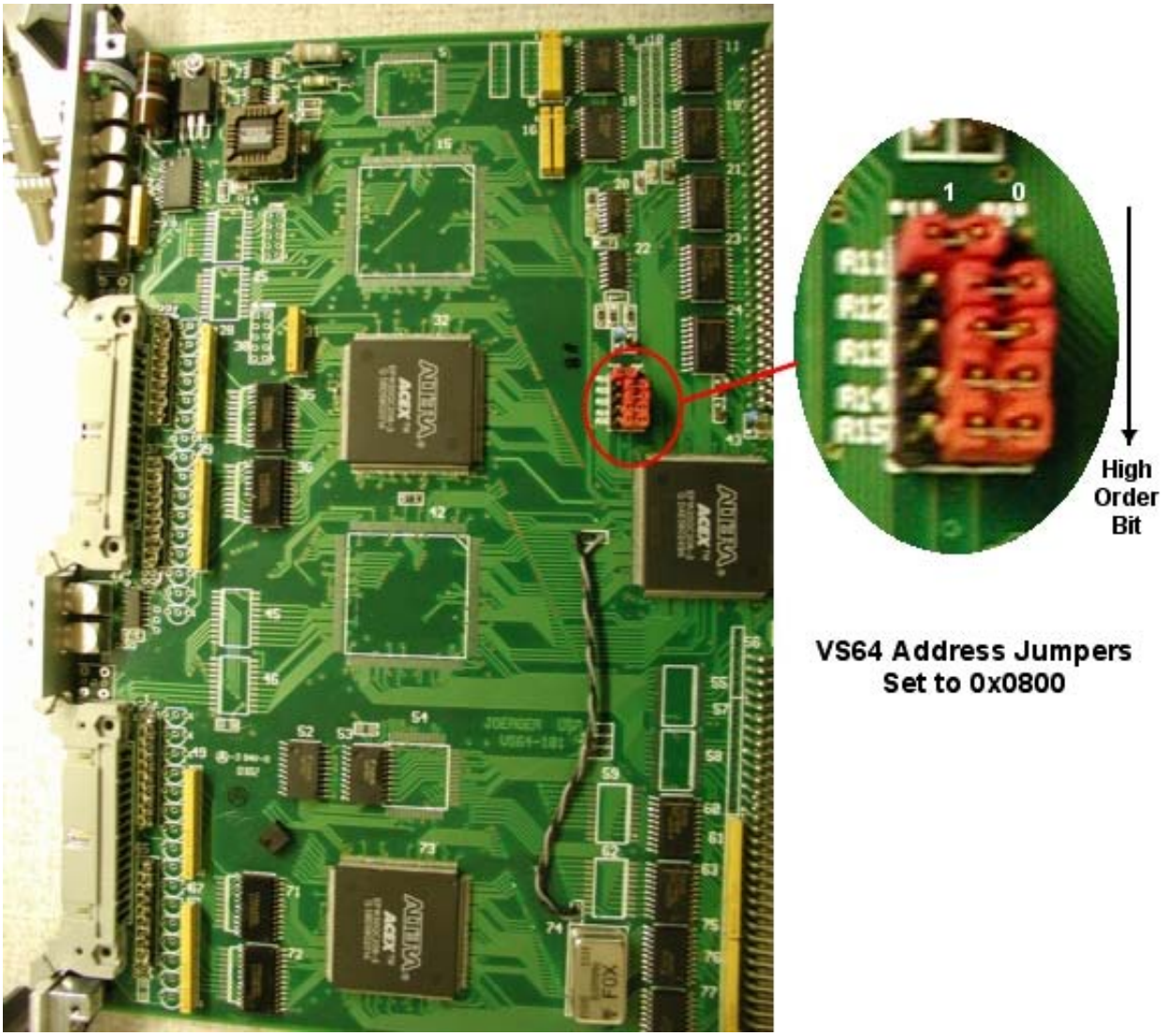


Figure 34
Jumper Settings for the Joerger VS64 Counter Module

7.1.3 Calibration Procedure

When using the "Two Gate" method for measuring the ring revolution period, the system needs to be calibrated to account for cable delays and gate delays within the VS64 and the V124S. The following procedure can be used to calibrate the measurement:

Equipment Required:

- A very stable, calibrated, 50 Mhz or greater, waveform or pulse generator such as the Agilent 81101A Pulse Generator.
- A calibrated 225 Mhz or greater counter such as the Hewlett Packard 53131A Universal Counter.

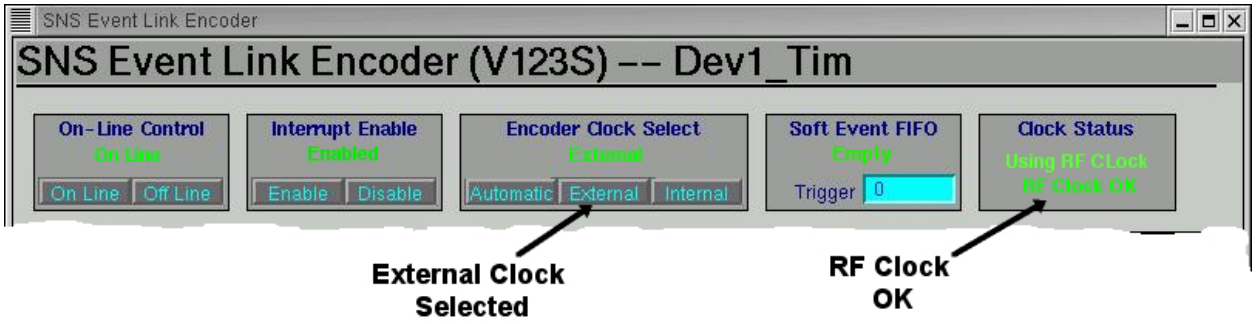
Setup:

- Set the waveform generator to generate a 33.85 Mhz sine wave (or 50% duty square wave) with zero offset and 2 volts peak to peak (±1 volt maximum). Use the "Internal PLL" setting to get the signal as stable as possible.

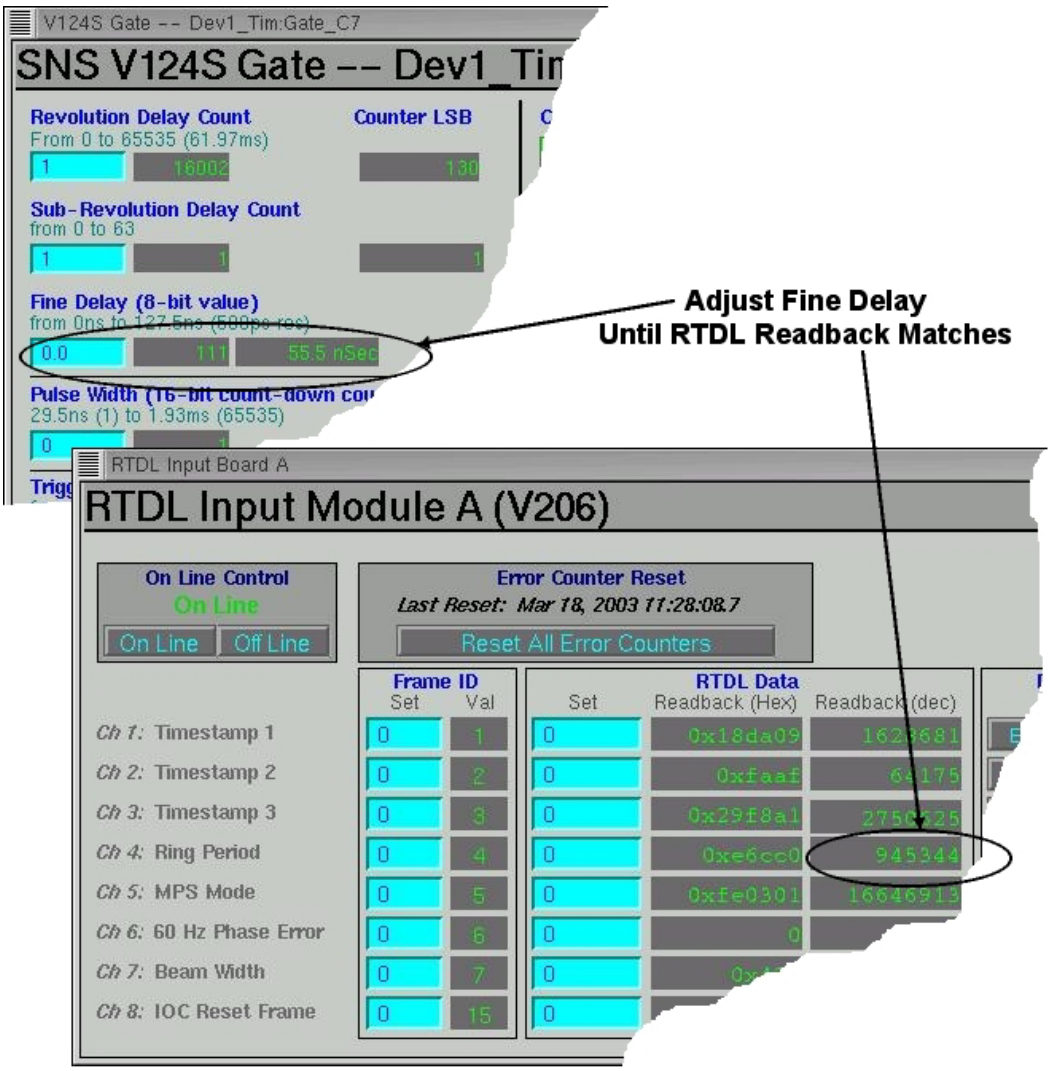
- Connect the output signal of the waveform generator to the “RF Clock In” port on the V123S event link encoder module.
- Connect the “Trigger Out” port of the waveform generator to the “Input” port of the counter.
- Set the counter to display frequency.

Procedure:

1) From the Timing Master EDM screen, call up the “Timing System Diagnostics” display. From this display, select the “Event Encoder (V123S) Screen”. Make sure that the “Encoder Clock Selection” is set to “Internal” and the“Clock Status” indicates that the module is using the RF clock and that the RF clock status is “OK”.



2) From the “Timing Diagnostics” screen, call up the V206 diagnostic display that contains the “Ring Period” RTDL frame (frame 4).
3) From the “Timing Diagnostics” screen, call up the diagnostic display for the V124S channel that provides the “Counter Transfer” gate (the second gate).
4) Read the measured output frequency from the counter module display. Hopefully this value should be stable to 7 or 8 digits. (e.g. 33.850145 Mhz).
5) Divide this number by 32 to get the ring revolution frequency (e.g. 1.057817 Mhz).
6) Take the inverse of this number and multiply by 1,000,000 to get the ring revolution period in picoseconds (e.g. 945343)
7) Adjust the fine delay of the “Counter Transfer” gate until the value displayed in the “Ring Period” RTDL frame matches the value computed above to within ±1 picosecond (see picture below):



7.2 SOFTWARE

The EPICS driver for the VS64 resides in the directory:

```
$IOCTOP/timingMaster/production/timingLib/src/drvVS64.c
```

It is built into the "timingMasterLib" library which is loaded by the timing master IOC. The header file "drvVs64.h" (in the same directory) contains the interface definitions and is installed in the directory:

```
$IOCTOP/timingMaster/production/include
```

The following routines are used by the timing master IOC:

initVS64 (baseAddr)

This routine is called by the timing master IOC's startup command procedure. It initializes the VS64 module.

Arguments:

baseAddr = A16 base address for the VS64 card.

Other Routines TBS.

8 TIMING GATE GENERATOR (V124S)

8.1 HARDWARE

The V124S card provides eight independently configurable timing channels which can be triggered from:

- An event on the SNS event link,
- An external TTL signal,
- A software trigger from the IOC.

Channels may also be combined to provide extra long delays or widths. The board can also interrupt the IOC on error conditions, channel termination conditions, and timestamp conditions.

For specific details about the V124S hardware, register map, and programming, see reference [3][*].

8.1.1 Board Configuration Properties

Each V124S card has several "Board Configuration" properties – properties apply to the card in general rather than individual timing channels. These properties include the card-wide delay, the revolution counter re-synch event, the timestamp re-synch event, the event link error counters, and the interrupt controls.

Figure 40 in section 8.2 shows the diagnostic EDM screen that displays the card-specific properties of the V124S module.

8.1.1.1 Board Command Register

The card as a whole can be enabled or disabled. When disabled, no gates are generated. For testing purposes, the card can also be configured to use an internal clock instead of the event link clock.

8.1.1.2 Interrupt Sources

The card can be configured to interrupt on any of the following sources:

- Event Link carrier error (loss of carrier signal)
- Event Link frame error
- Event Link parity error
- Timestamp re-synch event
- Individual channel termination
- Individual channel timestamp.

The SNS V124S driver software normally enables only the three event link error interrupts. To prevent a bad event link from flooding the system with interrupts, the interrupt rate is throttled down to 4 Hz maximum.

Timestamp re-synch and individual channel timestamp interrupts are not currently implemented. The timing master sequencer task uses the individual channel termination interrupt. The timing master IOC defines an "End-of-Cycle" gate (not an event) that occurs 100 turns after the Extract event. The timing master sequencer task waits on the termination interrupt of this gate to set up the RTDL and the variable rep-rate gates for the next cycle.

[*] The documentation in [3] may still contain some early design information that may be either superceded or not fully implemented. Where information in [3] conflicts with information contained in this document, this document will take precedence (except in those instances where I made a mistake).

8.1.1.3 Error Counters

Error counters are maintained for each of the three event link error conditions. These counters are useful for determining whether the event link has had problems in the past or is currently having problems. Since the error interrupt is throttled to 4 Hz maximum, these counters will only increment at a maximum rate of 4 Hz. The error counters are reset by an IOC reboot and by a "Clear Errors" call to the V124S driver.

8.1.1.4 Revolution Frequency Re-Synch Event

As mentioned in section 1.1.2, the event link clock is derived from the ring RF signal multiplied by 32. The timing system hardware needs to know which of the 32 × Frev clock pulses corresponds to the start of a ring revolution. This is the purpose of the "Revolution Frequency Re-Synch Event" register. When the V124S detects the event number contained in this register, it resets its revolution counter and declares the corresponding clock cycle (and every 32nd clock cycle afterward) to be the start of a revolution.

For all SNS applications, the value in this register should always be "1" (the Cycle-Start event).

8.1.1.5 Timestamp Reset Event

The value in this register should also always be "1" (the Cycle-Start event). This allows the V124S hardware to reset the timestamp counters at the start of each machine cycle. For more on time-stamping, see section 8.1.2.9 below.

8.1.1.6 Clock Delay

This register allows you to tweak the timing of the entire card by up to 127.5 nanoseconds, in increments of one half nanosecond (500 picoseconds).

8.1.2 Channel Configuration Properties

Each individual timing gate has its own associated delay, width, trigger, and timestamp. The delay contains coarse, medium, and fine controls. "Coarse control" is the "Revolution Delay Count", which aligns the gate with the start of a revolution and increments in units of "Turns" (about 945 nanoseconds at 1 GeV). "Medium control" is the "Sub-Revolution Delay Count", which increments in units of "sub-revolutions" (about 29.5 nanoseconds at 1 GeV). "Fine control" is in absolute units of 500 picoseconds.

The gate delay and width countdowns are written to buffer registers. The actual counters are loaded from these buffer registers on the gate's "Halt Trigger". The halt trigger occurs when the last output pulse from a triggered gate completes. The counters are also loaded immediately after the buffer registers are written so that the new values will be there the next time the gate is triggered. As a result of this, if the buffer registers are written while one of the counters is running, the counter register will be reloaded and the actual delay or width for that pulse will be unpredictable. If this is a problem, you should ensure that the delay and/or width buffer registers are not written while the gate is active.

Figure 41 in section 8.2 shows the diagnostic EDM screen from which a gate's individual properties can be displayed and adjusted.

8.1.2.1 Revolution Delay Count

This value specifies the number of revolutions (or "Turns") to delay between the time the gate is triggered and the first output pulse is produced. Recall that one revolution is 32 sub-revolutions. This does not mean, however, that a revolution delay count of 1 is always 945 nanoseconds. The counter is decremented on every "Revolution" clock pulse – which is determined by the Revolution Frequency Re-Synch event as described above in section 8.1.1.4. A non-zero revolution delay count will therefore cause the output gate to be synchronized with the start of a ring revolution – regardless of when the gate was triggered.

The revolution delay count register may have any value between 0 and 65535. When running with a revolution delay count of 0, however, care must be taken to ensure that the "Delay Control Register" specifies "Manual" for the revolution delay and either "Manual" or "Event" for the sub-revolution delay. Normally, the V124S driver software will set these values correctly for you (see section 8.1.2.8 below for a discussion of the delay control register).

8.1.2.2 Sub-Revolution Delay Count

This value specifies the number of sub-revolutions to delay after the revolution delay count expires. The sub-revolution delay must be between 1 and 63 (just under two revolutions). The V124S will not support a sub-revolution delay of zero. If you need to start a gate just at the start of a revolution, you will need to adjust the revolution delay counter to be one less than the delay you want and then set the sub-revolution delay counter to 32.

8.1.2.3 Fine Delay Count

This register allows you to do fine adjustments on the gate delay. The fine delay is not tied to the ring RF. The adjustment is in fixed time units of 500 picoseconds. The fine delay register can be set to any value between 0 and 127.5 nanoseconds.

8.1.2.4 Pulse Width

The pulse width is given in sub-revolutions. The value may be anywhere from 0 to 65535 sub-revolutions for a maximum gate width of about 1.9 milliseconds at 1 GeV. Note that a gate width of 0 will not produce an output pulse.

8.1.2.5 Trigger Count

This register determines how many pulses will be generated each time the gate is triggered. This is a 32-bit register, so the value can be anywhere from 1 to 4,294,967,295. The V124S driver does not allow a trigger count of 0 (which is really 4,294,967,296).

If the trigger count is greater than one, the revolution, sub-revolution, and fine delays are re-applied and another pulse is output. This continues until the trigger count is exhausted. Note that multiple pulses will only be generated if the revolution delay counter (section 8.1.2.1) is greater than zero.

It is important to remember that the revolution delay counter decrements on every "Revolution Clock" pulse – which is determined by the Revolution Frequency Re-Synch event (see section 8.1.1.4). Consequently, a "delay" of one revolution will not always translate into a delay of 945 nanoseconds. For example, Figure 35 below shows a trigger count of two with a revolution delay of 1 and a sub-revolution delay of 1. The second pulse occurs exactly 945 nanoseconds after the start of the first pulse, and one sub-revolution after the start of the next turn.

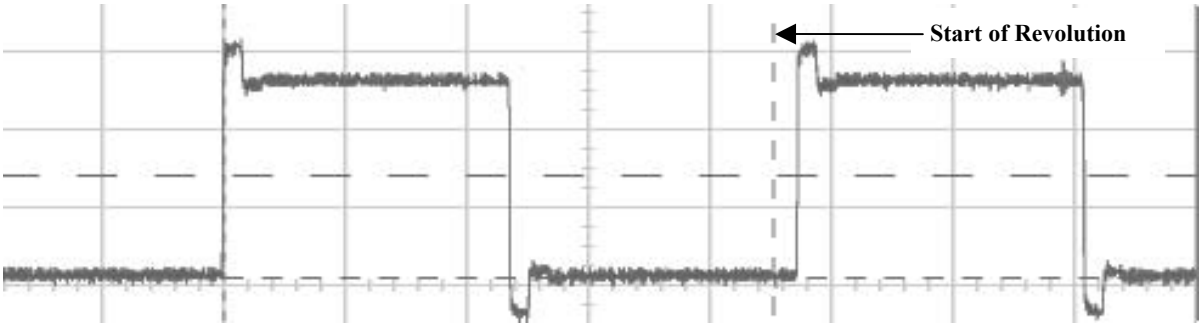


Figure 35 Multiple Triggers
Delay = 1 Revolution + 1 Sub-Revolution, Width = 16 Sub-Revolutions

In Figure 36, we see the same setup with a width of 24 sub-revolutions. Note that even though the width of the pulse has increased, the starting point of the second pulse stays the same.

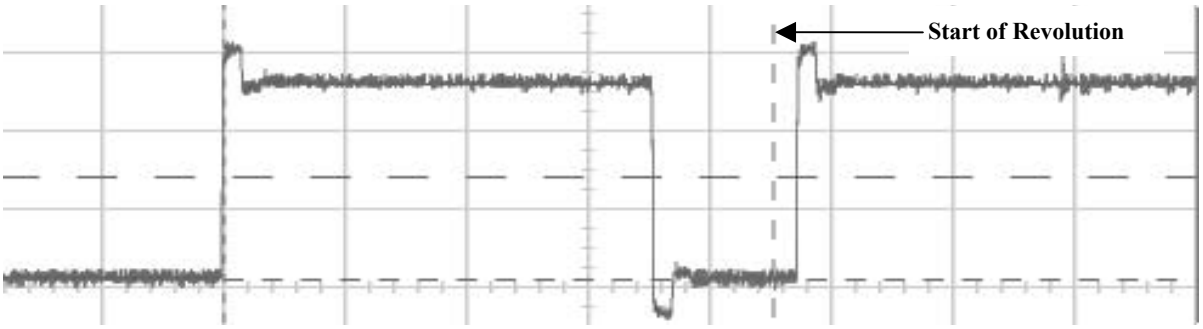


Figure 36
Width Increased to 24 Sub-Revolutions

There is a discontinuity when the width is 32. Instead of waiting until the start of the next revolution after the end of the first pulse (i.e. every other pulse), the second pulse begins after a delay of only 1 sub-revolution after the end of the first pulse.

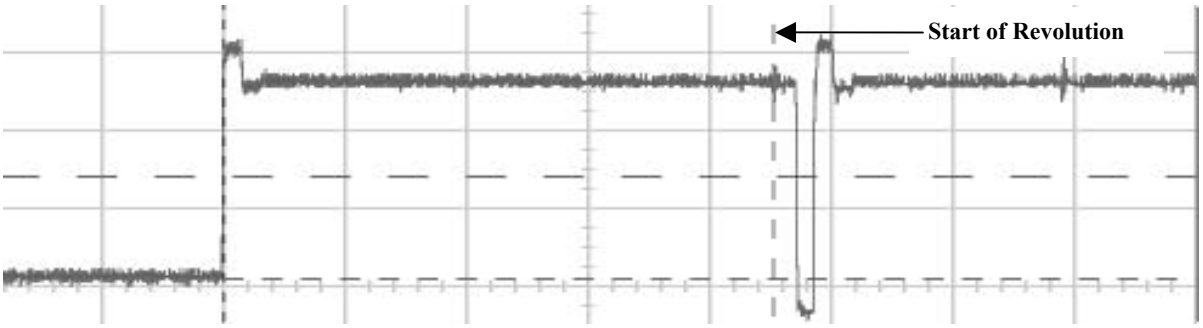


Figure 37
Width Increased to 32 Sub-Revolutions

If the trigger count is greater than 2, you will see an oscillating pattern of short and long delays between the pulses.

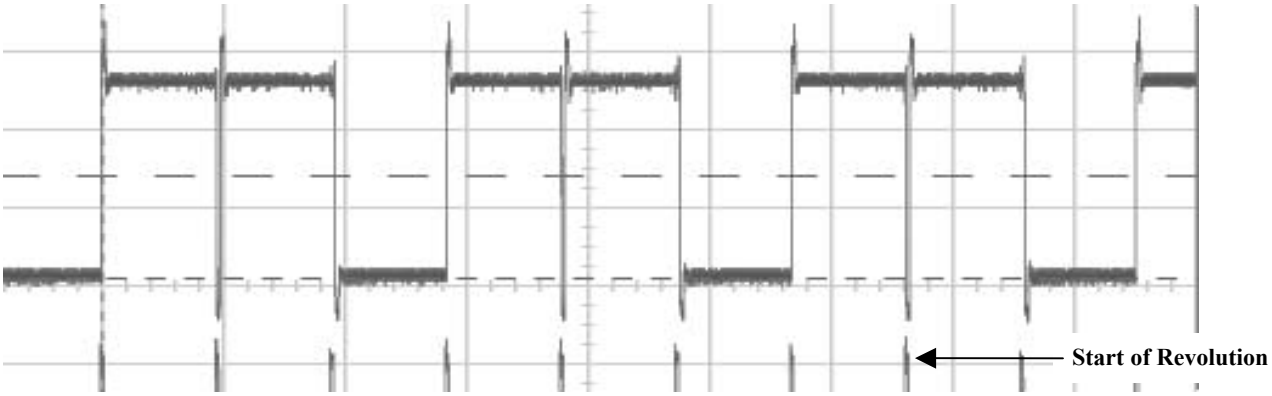


Figure 38
Multiple Trigger Pattern Anomaly When Width is 32 Sub-Revolutions

One final point to make about the trigger count is that because of the way revolutions are counted, each of the pulses will occur at the same point relative to the start of the turn (i.e. start plus one sub-revolution, start plus two sub-revolutions, etc.) This implies that if the revolution delay is one, and the sub-revolution delay and width are less than 32, there will always be exactly 32 sub-revolutions between the start of each pulse (excepting the "Width=32" anomaly). If the width or sub-revolution delay is between 33 and 63, there will always be 64 sub-revolutions (two turns) between each pulse.

8.1.2.6 Trigger Event

This register contains the number of the event that will trigger this gate. A value of zero means that the gate is not triggered by an event. When a non-zero number is written to this register, the V124S driver software automatically changes the values in the "Delay Control Register" (see section 8.1.2.8 below) to reflect the fact that the gate is triggered by an event.

It should be noted here that the V124S hardware is capable of triggering a gate from several different events at once. The SNS V124S driver software does not support this option.

8.1.2.7 Channel Control Register

This register allows you to set the output polarity of the pulse, stop the pulse from counting, reset the gate to a default state, trigger the pulse manually, and set it up for "Single-Shot". The "Trigger" button on the diagnostic screen shown in Figure 41 (section 8.2) will cause the gate to fire immediately.

"Single-Shot" mode is achieved by setting the "Reload Counters" bit to "Manual". A single-shot is triggered by writing the "Trigger Count" register.

8.1.2.8 Delay Control Register

The "Delay Control Register" determines when the revolution and sub-revolution delays get applied, and when the gate "Halts" (stop's producing output pulses). The register has three sections:

Options for the revolution delay count are:

Manual: Wait for a software trigger to start the revolution delay counter. This is the option used for variable rep-rate and single-shot gates. This option is also used when the revolution delay count is zero.

Event: Wait for the specified event to start the revolution delay counter.

External: Wait for an external TTL signal to start the revolution delay counter.

Previous: Wait until the previous channel's revolution delay has completed before starting this channel's revolution delay counter. This feature can be used to gang two channels together to produce very long delays.

Options for the sub-revolution delay count are:

Manual: Wait for a software trigger to start the sub-revolution delay counter. Do not wait for the revolution counter to finish. This option is normally only used when the revolution delay counter is 0.

Event: Wait for the specified event to start the sub-revolution delay counter. Do not wait for the revolution counter to finish. This option is normally only used when the revolution delay counter is 0.

Rev Done: Wait for the revolution counter to finish before starting. This is the option that should be used when there is a non-zero revolution delay.

Rev Start: Pretty much the same thing as "Rev Done". The V124S driver software uses "Rev Done" by default.

Options for the halt delay selection are:

No Halt: Channel never "Halts". Trigger count is ignored. Triggers are repeated *ad infinitum*.

Fine Delay: Gate "Halts" after the fine delay of the last pulse.

Last Pulse: Used to gang gates together for long delays

8.1.2.9 Time-Stamping

A V124S timing channel can be time-stamped either by the event link clock, or by specific events. The "Timestamp Configuration Control Register" controls how channel time-stamping is performed. This register has two sections. The first section controls when the timestamp is applied and the second section controls the clock used to measure the timestamp.

Four options are possible for when the timestamp is applied:

None: Time-stamping is not performed for this channel.

Event: Timestamp is applied when a specified event is detected. The number of the event that will trigger the timestamp is given in the "Timestamp Trigger Event" register.

First Pulse: Timestamp is applied at the start of the gate's first output pulse. This is the default value for channel time-stamping.

Last Pulse: Timestamp is applied at the start of the gate's last output pulse. If the trigger count is set to 1 (i.e. there only is one output pulse per trigger), the applied timestamp will be the same as in the "First Pulse" case.

There are two options for the timestamp clock:

Carrier: The 32 × Frev clock from the event link. When this option is selected, the timestamp value will be the number of subrevolutions from the last Cycle-Start event until the timestamp trigger.

Event: When this option is selected, the event number specified in the "Timestamp Clock Event" register provides the timestamp clock. The timestamp value will be the number of times the specified event occurred after the last Cycle-Start event. The timing master sequencer program uses this feature on the "End-of-Cycle" gate. By setting the timestamp clock event to "3" (MPS Auto Reset Event), the timing master can count the number of "Fast Protect" trips occurred during the last cycle and set the appropriate bit in the "Last Cycle Veto" RTDL frame.

When the event link clock is used to timestamp a channel, the V124S hardware will always return a value that is slightly larger than the actual timestamp. The exact amount is determined by whether you trigger on the first or last output pulse.

- If you trigger on the first pulse, the timestamp value will be two subrevolutions too large.
- If you trigger on the last pulse, the timestamp value will be three subrevolutions too large.

The V124S driver software compensates for these offsets, so that the timestamp value will always appear correct.

8.1.3 Jumper Settings

The VME A16 base address is set via five jumpers located in the upper middle of the card (see Figure 39 below). The jumpers set base address lines A11 - A15 (2K boundary). With the board standing upright (as shown below), the high-order bit is at the top and the low order bit is toward the bottom. If a jumper is present, it represents a logical "0". If the jumper is removed, it represents a logical "1".

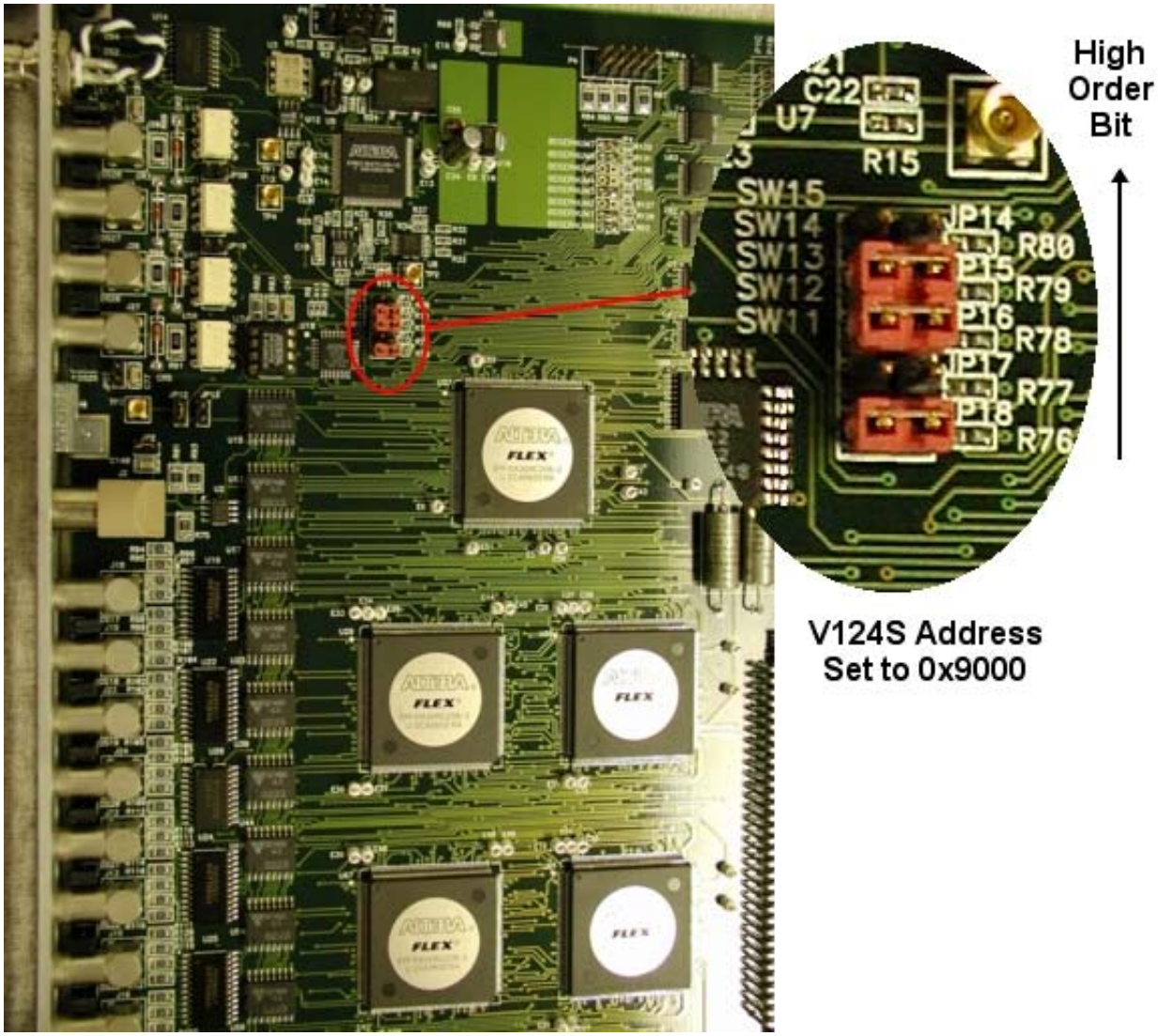


Figure 39
V124S Address Selection

8.2 DIAGNOSTIC SCREENS

The top level V124S diagnostic screens can be obtained by starting at the Timing Master EDM screen and selecting the "Timing System Diagnostics" related display button. From the Timing System Diagnostic screen, select the "Event Trigger (V124S) Screens" related display button. This will bring up a menu from which you can select the top-level screen for the V124S card you wish to observe:

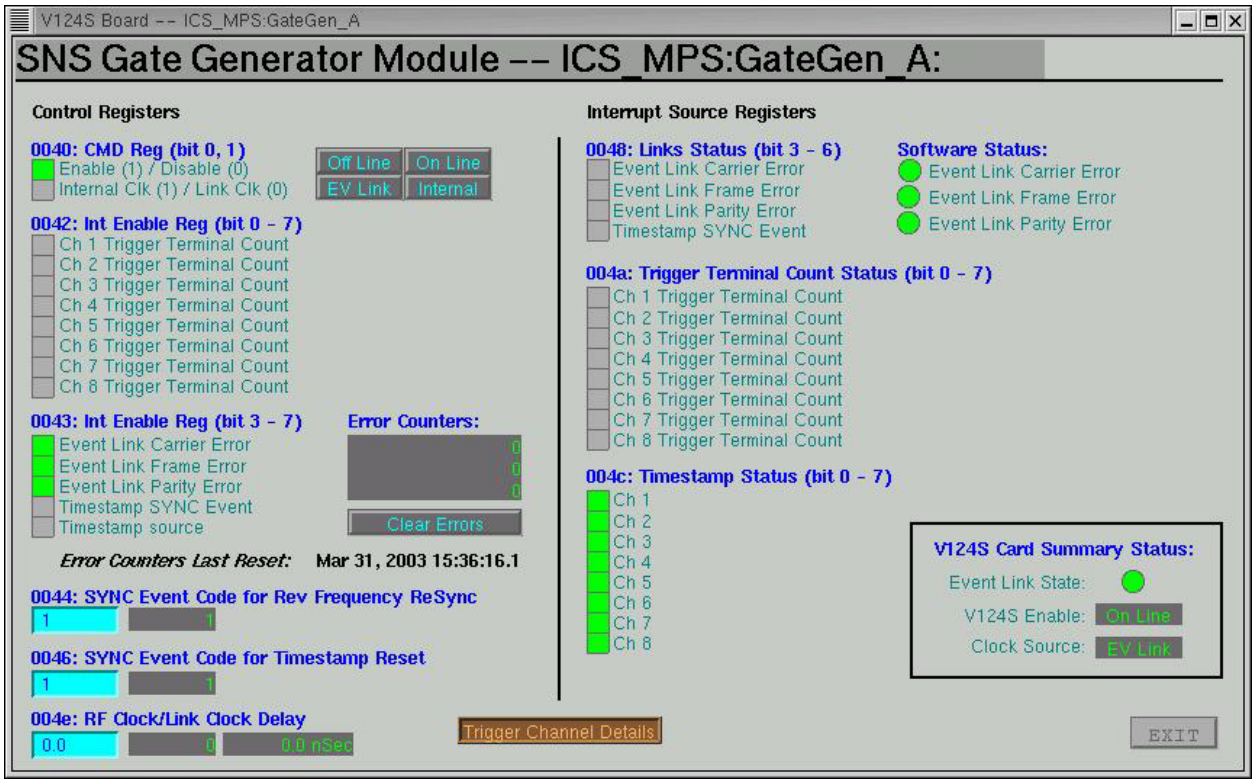


Figure 40
Top Level V124S Diagnostic Screen

In a normally operating V124S card, the CMD register (upper left corner) should indicate "Enabled" (bit on) and "Link Clk" (bit off). The "Trigger Terminal Count" interrupts are normally not enabled. The "Event Link" error interrupts should be enabled and the related error counters should not be incrementing. The "SYNC Event Codes" for both the "Rev Frequency ReSync" and the "Timestamp Reset" should be 1.

To see the details about an individual V124S gate, select the "Trigger Channel Details" menu button on the bottom of the screen and chose the desired channel card you wish to view. This will bring up the following screen:

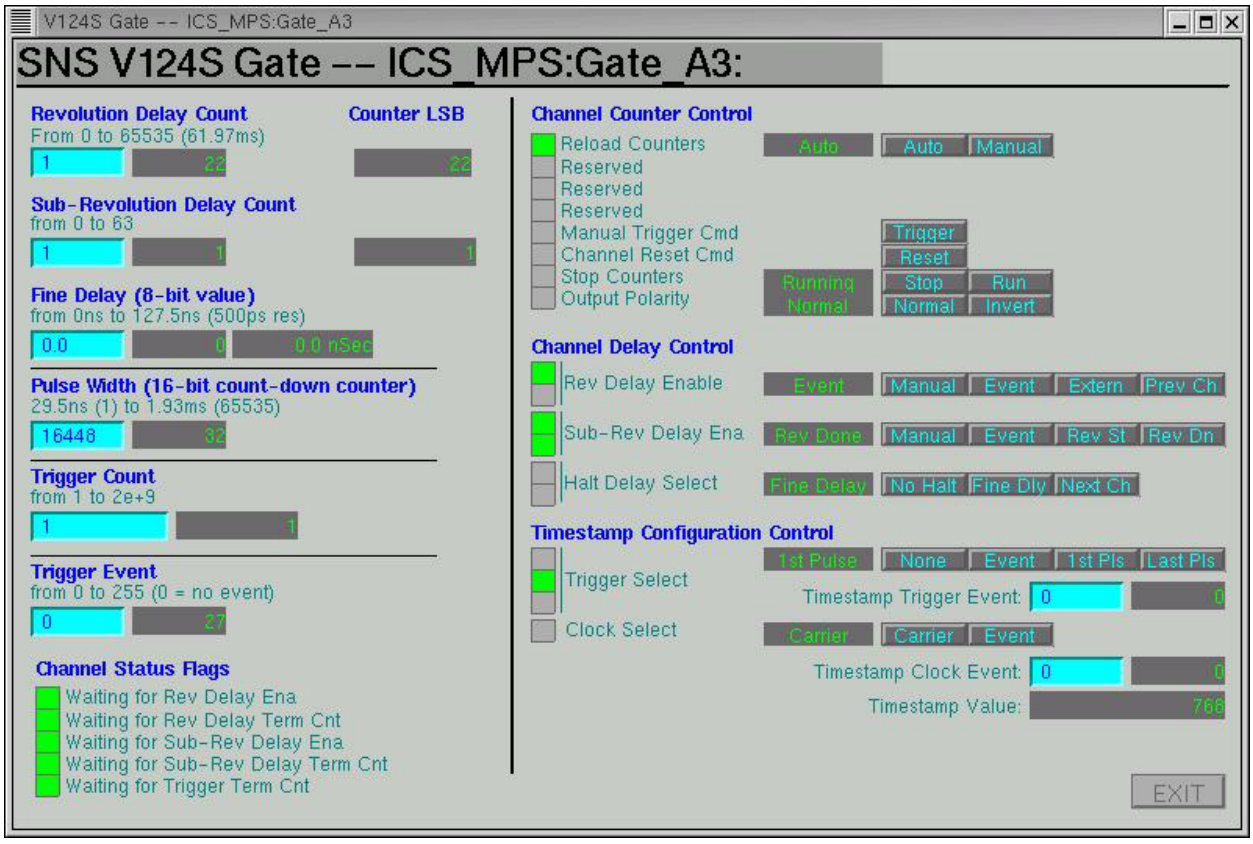


Figure 41
V124S Individual Gate Diagnostic Screen

The values in the light blue boxes are setpoints. The values in the dark gray boxes are readbacks. As can be seen above, the setpoint values may not necessarily reflect the readback values – particularly if the IOC has been through a "power-on" reboot.

8.3 EPICS SOFTWARE

The EPICS software for the V124S module resides in the directory:

```
$SHARE/timing/production/
```

The “bin”, “db”, “dbd”, “include”, and “opi” directories all contain useful items. The “bin” directory contains architecture-specific sub-directories, each of which contain the file:

```
snsTimingLib
```

This file contains the EPICS driver and device support for the V124S module. It also contains the subroutines for implementing variable rep-rates and for translating between timing system units (turns, sub-revolutions, fine-delay) into clock time (microseconds).

The “dbd” directory contains the following .dbd files:

- **timingGateInclude.dbd:** This is an unexpanded dbd include file that references all the record types and all the driver and device definitions needed by the database templates in the “db” directory. This file is suitable for inclusion in an application’s “*xxx*AppInclude.dbd” file.
- **timingGate.dbd:** This file only includes the V124S driver and device definitions. This file is suitable for loading at boot time, via a “dbLoadDatabase” call, if your IOC does not build a single application-wide .dbd file.
- **timing.dbd:** This file is a fully expanded version of “timingGateInclude.dbd”. This file is primarily used by vdct as the reference .dbd file when constructing or editing the template files.

The “db” directory contains useful template files, including the diagnostic templates for individually inspecting the specifics of each gate. The diagnostic templates (and their macros) are:

v124s.template

Diagnostic database for reading and controlling all the "card-wide” registers of the V124S module. These include clock selection, card-wide delay, error counters, etc. This template should be instantiated once for each V124S card in the system.

Macros:

S = System Name
SS = Sub-System Name
N = System Instance
DI = Device Instance -- typically the board "letter" (A, B, C, ...)
CD = Card number (0, 1, 2, ...)

The above macros are used to construct the PV names, which are of the form:

$(S)_$(SS):GateGen$(N)_$(DI):<signal>

A typical PV name using this format might be:

ICS_Tim:GateGen_A:Enable

(note that in this case the system instance, $(N), is empty.)

timingGateDiag.template –

Diagnostic database for reading and controlling all the channel-specific registers of the V124S module. This template should be instantiated once for each of the eight channels on a V124S card

Macros:

S	=	System Name
SS	=	Sub-System Name
N	=	System Instance
DI	=	Device Instance -- typically the board "letter" followed by the gate “number” (e.g. A1, A2, A3, …)
CD	=	Card number (0, 1, 2, ...)
GATE	=	Gate number (1, 2, ... 8)

The above macros are used to construct the PV names, which are of the form:

$(S)_$(SS):Gate$(N)_$(DI):<signal>

A typical PV name using this format might be:

ICS_Tim:Gate_A1:FineDly

(note that in this case the system instance, $(N), is empty.)

Other useful template files are:

staticGate.template –

Database template for gates that are fixed in time and do not have operator-adjustable parameters (hence the term “static”). The gate's parameters are all specified as macros to the template. The generated records all have "PINI=YES", so that the gate parameters are written once at boot time.

Macros Defining the PV Names:

S	=	System Name
SS	=	Sub-System Name
N	=	System Instance
DI	=	Device Instance -- typically the gate's "name"

Macros Defining the Gate's VME Address

CD	=	Card number (0, 1, 2, ...)
GATE	=	Gate number (1, 2, ... 8)

Macros Defining the Gate's Parameters:

RevDly	=	Gate's revolution delay
SubRevDly	=	Gate's sub-revolution delay (0 - 63)
FineDly	=	Gate's fine delay (in nanoseconds)
Width	=	Gate width (in Sub-Revolutions)
Event	=	Event number to trigger the gate from. 0 = no event.
TrigCnt	=	Number of gates to generate per trigger (usually 1)
Polarity	=	0 means normal polarity (high true) 1 means reverse polarity (low true)
AutoReload	=	1 means gate is automatically triggered 0 means the gate is manually triggered by VME command or by "single-shot"

dataTrigger.template –

Database template for data acquisition trigger gates. These gates have operator-adjustable delays (turns, sub-revolutions, fine delay) which are referenced relative to the start of the beam gate. The operator may also select whether to trigger the gate on the "Fast Trigger" (6 Hz), the "Slow Trigger" (1 Hz), the "Snap Shot Trigger" (triggered manually), or on no trigger at all. The gate width, trigger count, polarity, and AutoReload flag are all configured by macros in the template.

Macros Defining the PV Names:

S	=	System Name
SS	=	Sub-System Name
N	=	System Instance
DI	=	Device Instance -- typically the name of the triggered device

Macros Defining the Gate's VME Address

CD	=	Card number (0, 1, 2, ...)
GATE	=	Gate number (1, 2, ... 8)

Macros Defining the Gate's Parameters:

Width	=	Gate width (in Sub-Revolutions)
TrigCnt	=	Number of gates to generate per trigger (usually 1)
Polarity	=	0 means normal polarity (high true) 1 means reverse polarity (low true)
AutoReload	=	1 means gate is automatically triggered 0 means the gate is manually triggered by VME command or by "single-shot"

variableRepRate.template –

Database template to set up a gate with a variable rep-rate. This template does not affect any of the gate's parameters other than rep-rate. The rest of the gate parameters will need to be set up via another method (such as the staticGate.template).

Macros Defining the PV Names:

S	=	System Name
SS	=	Sub-System Name
N	=	System Instance
DI	=	Device Instance -- typically the name of the triggered device

Macros Defining the Gate's VME Address

CD	=	Card number (0, 1, 2, ...)
GATE	=	Gate number (1, 2, ... 8)

Macros Defining the Gate's Parameters:

OFFSET	=	The offset (in cycles) from the constraint pattern. This is useful for creating "precursor" gates. (OFFSET = -1)
BASE	=	The constraining rep-rate pattern. This will typically point to the VALB field of a rep-rate gensub record (see section 8.5.1). Note that the macro should specify the both the record and field names. For example: "ICS_Tim:Gate_SourceOn:RRSub.VALB" If the BASE macro does not specify a valid PV name, then the rep-rate pattern will be unconstrained. Typically, an unconstrained rep-rate is specified by setting BASE = 60 (although any number would do).

MAXREP = Maximum rep-rate value.
SOFT = If 1 (TRUE), the gate should not be triggered on the cycles it is scheduled for. It should only be marked as triggered.
If 0 (FALSE), trigger the gate on the cycles it is scheduled for (this feature is used by the timing master sequencer for beam-related gates that the sequencer wants closer control of).

8.3.1 Setting Up a Timing Client

The V124S software support depends on the gensub record and the SNS Utility Module support libraries. Client IOCs that use the V124S card must therefore link with and load the gensub record support, the utility module software, and the V124S software. The following areas in the client IOC application may need modification:

8.3.1.1 config Directory RELEASE File

The first step in adding a V124S card to your IOC is to go to the top level "config" directory, and edit the RELEASE file. Make sure the following symbols (or their equivalent) are defined:

```
GENSUB  = $(SHARE_RELEASE)/Gensub
UTILITY = $(SHARE_RELEASE)/utility/production
TIMING  = $(SHARE_RELEASE)/timing/production
```

This will make sure that the other application Makefiles have access to all the gensub, utility module, and V124S definitions and code they need.

8.3.1.2 src Directory Makefile.Vx File

The SNS standard is for an application to copy all the system and utility software it needs into its local "bin" directory and load it from the local directory at boot time. If your IOC implements this method, you will need to edit the "Makefile.Vx" file in your application's "src" directory and add the following lines:

```
BIN_INSTALLS += $(GENSUB_BIN)/genSubRecord.o
BIN_INSTALLS += $(UTILITY_BIN)/snsUtilLib
BIN_INSTALLS += $(TIMING_BIN)/snsTimingLib
```

These lines will cause a "make" in the "src" directory to copy the latest versions of the gensub, utility module, and timing client libraries into your local "bin" directory.

8.3.1.3 src Directory *xxx*AppInclude.dbd File

A good practice is to create an application-wide ".dbd" file. The gensub, utility, and timing directories all contain "*xxx*Include.dbd" files that contain all the record, device, and driver definitions you need for each library. Any duplicate definitions will get sorted out when the *xxx*AppInclude.dbd file is expanded into the *xxx*App.dbd file during the make process (note that duplicate .dbd definitions are not sorted out when several .dbd files are loaded at boot time – this is why the *xxx*AppInclude.dbd method is preferred).

To make sure all the relevant timing, utility and gensub definitions are included in your application's .dbd file, insert the following lines into your *xxx*AppInclude.dbd file:

```
include "snsUtilInclude.dbd"
include "timingGateInclude.dbd"
```

8.3.1.4 src Directory base.dbd and baseLIBOBJS Files

If you are using the *xxx*AppInclude.dbd method to build an application-wide .dbd file, then the only thing you need to worry about including in the "base.dbd" file are the base record, device, and driver support routines that your own application uses.

The baseLIBOJBS file, on the other hand should be edited to include all record types used by the timing and utility module databases. These record types are:

```
aiRecord        aoRecord        biRecord        boRecord
calcRecord      calcoutRecord   eventRecord     longinRecord
longoutRecord   mbbiRecord      mbboRecord      selRecord
subRecord       stringinRecord
```

8.3.1.5 Db Directory *xxx*App.substitutions File

Each IOC with a V124S card should include the basic V124S diagnostic databases. This will allow for easier troubleshooting of timing system problems. The V124S diagnostic database consists of two parts, a card-specific part, and a channel-specific part. The card-specific part contains overall status information about the V124S card. It is represented by the template file, "v124s.template". The channel-specific part contains information about the individual timing channels. It is represented by the template file "timingGateDiag.template".

If you use a substitutions file to create your databases, you can define the V124S diagnostic database here with lines of the form:

```
#---------------------
# Diagnostic Records for the V124S card
#
file v124s.template {
    {S=<<system>> SS=<<sub-system>>, N=<<system-instance>>, DI=A, CD=0}
}

#---------------------
# Diagnostic Records for Each V124S Timing Gate
#
file timingGateDiag.template {
    {S=<<system>>, SS=<<sub-system>>, N=<<system-instance>>, DI=A1, CD=0, GATE=1}
    {S=<<system>>, SS=<<sub-system>>, N=<<system-instance>>, DI=A2, CD=0, GATE=2}
    {S=<<system>>, SS=<<sub-system>>, N=<<system-instance>>, DI=A3, CD=0, GATE=3}
    {S=<<system>>, SS=<<sub-system>>, N=<<system-instance>>, DI=A4, CD=0, GATE=4}
    {S=<<system>>, SS=<<sub-system>>, N=<<system-instance>>, DI=A5, CD=0, GATE=5}
    {S=<<system>>, SS=<<sub-system>>, N=<<system-instance>>, DI=A6, CD=0, GATE=6}
    {S=<<system>>, SS=<<sub-system>>, N=<<system-instance>>, DI=A7, CD=0, GATE=7}
    {S=<<system>>, SS=<<sub-system>>, N=<<system-instance>>, DI=A8, CD=0, GATE=8}
}
```

8.3.1.6 iocBoot Directory st.cmd File

The IOC startup command file will require several modifications. First, the gensub record, utility module, and timing support libraries must be loaded. If you load all your software from your local "bin" directory, then the following lines should be added to your st.cmd file:

```
cd appbin
   :
<< Other System Libraries >>
   :
ld < gensubRecord.o
ld < snsUtilLib
ld < snsTimingLib
```

Note that it is important to load "snsTimingLib" after loading "gensubRecord.o" and "snsUtilLib".

Next you need to initialize the utility module and the V124S hardware. Before you call "iocInit", you should have the following lines:

```
#-------------------------------------------------
# Initialize the Utility Module
#-------------------------------------------------

snsUtilInit 0, 0x4000
snsUtilConfigEvent 0, 0x40, 5

#-------------------------------------------------
# Initialize the V124S Timing Gate Generator
#-------------------------------------------------

v124sConfig 0, 0x9000, 0x3A, 3, 0.0
```

These lines initialize the SNS utility module and V124S cards with their default addresses, interrupt vectors, and interrupt request priorities.

If your IOC does fragmented dbd loads, the file "$SHARE/timing/production/dbd/timingGate.dbd" contains just the V124S driver and device definitions. The template files in "$SHARE/timing /production/db" can also be loaded at boot time via "dbLoadRecords".

8.4 DRIVER SOFTWARE

The EPICS driver and device support software for the V124S card reside in the files:

```
$SHARE/timing/production/timingApp/src/drvV124S.c
$SHARE/timing/production/timingApp/src/devV124S.c
```

These files are built into the "snsTimingLib" library which needs to be loaded by any IOC that uses a V124S card. The header file "v124s.h" (in the same directory) contains the interface definitions and is installed in the directory:

```
$SHARE/timing/production/include
```

The routines below are implemented in the drvV124S.c module.

status = v124sConfig (card, base, vector, level, offset)

This routine is called from the vxWorks startup file to configure the V124S card's VME address, interrupt vector, and base time offset.

Arguments:

card	=	Card number (as specified in the EPICS record INP/OUT link fields).
base	=	Base address in A16 space.
vector	=	Interrupt vector.
level	=	Interrupt request level.
offset	=	(double) Time offset adjustment for this card (in nanoseconds).

Returns:

OK	=	V124S card was successfully configured.
ERROR	=	V124S card configuration had errors.

status = v124sInit ();

This routine is normally called only from the EPICS iocInit routine. It completes the V124S card initialization by connecting the interrupt vectors, and enabling card interrupts.

Returns:

OK	=	V124S card was successfully initialized.
ERROR	=	V124S card initialization had errors.

v124sReport (level)

This routine is called from the EPICS dbior routine to report the location and status of each V124S card configured for the system. It may also be called from the vxWorks shell.

Arguments:

level	=	Level of detail to report. This parameter is currently ignored.

V124S_HANDLE = v124sCardHandle (card)

Returns a driver handle for the specified V124S card.
Returns NULL if the specified card was not configured (via the v124sConfig function).

Arguments:

card	=	Card number to return the handle for.

status = v124sGet (handle, channel, tag, &value)

This is the generic driver routine for returning scalar information from the V124S driver and hardware.

Arguments:

handle	=	Driver handle to the V124S card we want to read from.
channel	=	Channel number (1 - 8) we want to read from. Channel 0 is used to return card-specific (as opposed to channel-specific) data.
tag	=	Function code describing what we want to read (tag values are defined in v124s.h).
value	=	Address of an unsigned longword to receive the data.

Returns:

OK	=	The read succeeded.
ERROR	=	The read failed.

status = v124sGetBytes (handle, channel, tag, size, &buffer)

This is the generic driver routine for returning byte array information from the V124S driver and hardware.

Arguments:

handle = Driver handle to the V124S card we want to read from.
channel = Channel number (1 - 8) we want to read from. Channel 0 is used to return card-specific (as opposed to channel-specific) data.
tag = Function code describing what we want to read (tag values are defined in v124s.h).
size = Size of return buffer (in bytes)
buffer = Address of buffer to receive the data.

Returns:

OK = The read succeeded.
ERROR = The read failed.

status = v124sPut (handle, channel, tag, value)

This is the generic driver routine for writing scalar information to the V124S driver and hardware.

Arguments:

handle = Driver handle to the V124S card we want to write to.
channel = Channel number (1 - 8) we want to write to. Channel 0 is used to write card-specific (as opposed to channel-specific) data.
tag = Function code describing what we want to write (tag values are defined in v124s.h).
value = Unsigned longword containing the data to write.

Returns:

OK = The write succeeded.
ERROR = The write failed.

status = v124sWaitTrigger (handle, channel)

Wait for a "Halt Trigger" interrupt from the requested channel.

Arguments:

Handle = Driver handle for the V124S card.
channel = Channel to wait on.

8.5 VARIABLE REP-RATES

Variable rep-rates are implemented by putting the gate in "Single Shot" mode. A high-priority task triggered off the "RTDL-Valid" event, examines the gate's "rep-rate pattern" and determines whether or not to enable to gate for triggering on the upcoming cycle. The rep-rate pattern is computed by a gensub record using algorithms described in [8] and [9].

The software for implementing variable rep-rates resides in the file:

```
$SHARE/timing/production/timingApp/src/repRateUtil.c
```

This file is built into the "snsTimingLib" library which should be loaded by any IOC that uses a V124S card.

8.5.1 The Rep-Rate Gensub Record

Any gate with a variable rep-rate must have a "rep-rate gensub" record associated with it. The rep-rate gensub record computes the rep-rate pattern, makes sure that any dependencies are met, and keeps track of the gate for the high-priority task that triggers the gate on the cycles it is enabled for.

The rep-rate gensub record should be set up as follows:

Scan Parameters:

SCAN = Passive

Subroutine Names:

INAM =	v124sRepRateInit	(Name of Initialization Routine)
SNAM =	v124sRepRate	(Name of Processing Routine)
UFB =	v124sRepInfoSize	(Size of Input B array)
UFVB =	v124sRepInfoSize	(Size of VALB array)
UFVC =	v124sSuperCycleSize	(Size of VALC array)

Input Links:

INPA = Link to an ao record containing the rep-rate to setpoint.

INPB = Link to the rep-rate information structure for the constraining pattern. The rep-rate pattern of this gate will be constrained to match up with the rep-rate pattern specified by this link (after applying any offset). Typically, this field will point to the VALB field of another rep-rate gensub record. If there is no constraining pattern, then this field should be set to a constant (e.g. 0.0 or 60.0).

INPC = The VME address of the gate we are setting the rep-rate for. This value should be a two-digit number of the form, "sc", where "c" is the card number of the gate's V124S module, and "s" is the signal number of the particular gate. The signal number is specified first because card numbers can start with 0, and a leading zero will cause the record to interpret the value as an octal number.

INPD = Link to a bo record which is the Rep-Rate mode selector.
This value should either be "Fixed" (0), or "Variable" (1).
In "Fixed" mode, the gate's rep-rate is entirely controlled by the event it is triggered from.
In "Variable" mode, the rep-rate is controlled by both the triggering event and by the rep-rate

pattern.
This field is mostly for implementing variable rep-rates at the local (client) level. For example, a local RF IOC may want to alternate between generating local gates for conditioning, or using the default gates provided by the timing master IOC.

INPE = Constraining pattern offset.
When INPB points to the structure for a constraint pattern, this parameter determines where to place the generated pattern's cycles with respect to the constraining pattern. If the value is positive, the generated cycles must follow the constraint pattern by the specified amount. If the value is negative, the generated cycles must precede the constraint pattern by the specified amount. If the value is zero, the generated cycles must be coincident with the constraint pattern.

INPF = Soft Trigger Flag.
If TRUE, this value indicates that the gate should not be triggered on the cycles it is scheduled for. It should only be marked as triggered.
This field is used mostly for beam-related variable rep-rate gates in the timing master IOC. It allows the timing master sequencer task to decide whether or not to really fire a gate even though it is scheduled for that cycle.

Output Values:

VALA = (double) The actual rep-rate of the new pattern. This value may be smaller that the requested rep-rate (INPA) due to the constraint pattern.

VALB = (char array) The rep-rate information structure for the generated pattern (suitable for importing as a constraint pattern into the INPB field of another rep-rate gensub record).

VALC = (char array) The rep-rate bitmap array for the generated pattern.

Field Type Declarations:

FTA	=	DOUBLE	(Field type for Input A)
FTB	=	CHAR	(Field type for Input B)
FTC	=	USHORT	(Field type for Input C)
FTD	=	ENUM	(Field type for Input D)
FTE	=	LONG	(Field type for Input E)
FTF	=	CHAR	(Field type for Input F)
FTVA	=	DOUBLE	(Field type for Output A)
FTVB	=	CHAR	(Field type for Output B)
FTVC	=	CHAR	(Field type for Output C)

The initialization routine specified in the INAM field (v124sRepRateInit) performs a number of important initialization functions. It makes sure the record fields were set up correctly. If there a constraint pattern was specified (INPB field points to another rep-rate gensub record), it makes sure that the "Process Passive" and "Maximize Severity" options are set. This will guarantee that the constraint record will be processed before this record. It also adds this record to a linked list of "variable rep-rate" records where it can be found by the high priority "gate firing task" and the low-priority "rep-rate record processing task" (see section 8.5.4).

8.5.2 The Constraint Mux Gensub Record

Some times it is desirable to be able to select which other variable rep-rate gate you want your own gate to depend on. The "Constraint Mux" gensub record behaves like a regular rep-rate gensub record,

except that instead of computing a rep-rate pattern of its own, it allows you to select from one of up to 16 patterns computed by other rep-rate gensub records.

The constraint mux gensub record should be set up as follows:

Scan Parameters:

SCAN = Passive

Subroutine Names:

INAM	=	v124sConstraintMuxInit	(Name of Initialization Routine)
SNAM	=	v124sConstraintMux	(Name of Processing Routine)
UFB-UFQ	=	v124sRepInfoSize	(Size of Input B-Q array)
UFVB	=	v124sRepInfoSize	(Size of VALB array)

Input Links:

INPA = Link to a bo or mbbo record that provides the index for selecting which constraint pattern to use. An index value between 0 and 15 selects Input Links INPB, INPC, INPD, ..., INPQ.

INPB-INPQ = Links to the rep-rate information structures for the constraining pattern choices. These fields should either point to the VALB field of another rep-rate gensub record, or should be constants. The INPx field selected by the index (INPA) will be copied to the VALB field of this record. If the selected INPx field is a constant (and not a PV), the "unconstrained" pattern (all cycles are legal) is written.

Output Values:

VALA = (double) The actual rep-rate of the selected pattern.

VALB = (char array) The rep-rate information structure for the selected pattern (suitable for importing as a constraint pattern into the INPB field of a rep-rate gensub record).

Field Type Declarations:

FTA	=	ENUM	(Field type for Input A)
FTB-FTQ	=	CHAR	(Field type for Inputs B-Q)
FTVA	=	DOUBLE	(Field type for VALA)
FTVB	=	CHAR	(Field type for VALB)

Only those input fields that are actually pointing to other rep-rate gensub records should be given field types and widths (via the FVx and UFBx fields).

The initialization routine specified in the INAM field (v124sConstraintMuxInit) performs a number of important functions. It makes sure that each of the selectable input links is set up correctly. If an input link points to a rep-rate gensub record, it makes sure that the “No Process Passive” and “No Maximize Severity” options are set. This is the opposite behavior of the rep-rate gensub initialization routine. When a constraint mux gensub is processed, we only want to process the record connected to the selected input link.

8.5.3 Other Rep-Rate Related Routines

In addition to the gensub initialization, definition, and size computation routines described above, the RepRateUtil.c module defines the following other utility routines used for variable rep-rate processing.

repRateInfo* = v124sRepRateInfo (handle, gate)

Returns the address of the rep-rate information structure for the specified gate.
Returns NULL if the specified gate was not on the "variable rep-rate" list.
The repRateInfo structure is defined in v124s.h.

Arguments:
handle = Driver handle to the v124S card we want.
gate = Gate number (signal) of the desired gate.

v124sSetNewRepRates ()

Triggers the rep-rate gensub record scan task to begin computing new rep-rate patterns. This routine is normally only called inside the V124S "repRateUtil.c" package. It is externally available so that it may be called to initialize the rep-rates with saved values.

v124sTriggerVariableGates (cycle)

Triggers gates with variable rep-rates.
This routine expects to be called once per cycle. It looks at each of the variable-rep-rate gates and if a gate is scheduled to fire on the current cycle it is "armed". This routine also handles the bookkeeping chores involved in the process of setting a new set of rep-rate patterns. While the process of computing new patterns is underway (state = SETTING_NEW_RATES), this routine will check each time it is called to see if all the variable-rep-rate gates have finished computing their new patterns. When each gate contains "TRUE" in the "new" field of its rep-rate information structure, the routine sets each gate's bitmap array from the new pattern. That way all the gates are switched to the new pattern set simultaneously. The routine then clears the gate's "new" field, sets the status variable to "NEW_RATES_SET", and posts an EPICS I/O event to signal that the new patterns are now in use. The routine maintains the NEW_RATES_SET state for 10 seconds and then returns the state to IDLE.

Note: This routine should be called early enough before the start of the cycle (Cycle-Start Event) to ensure that all the scheduled gates for that cycle are armed in time. It should also be called late enough in the previous cycle to ensure that the gates are not still counting. To accomplish this, the routine is generally called right after the "RDTL-Valid" Event.

Arguments:
cycle = Current cycle number within the supercycle.

8.5.4 The Rep-Rate Computation Process

Computing new rep-rate patterns can be a fairly compute-intensive process, depending on if there are constraints and how well the requested rep-rates fit within the constraints. The details of the computation algorithms are given in [8] and [9]. There are 3 requirements for setting new rep-rate patterns:

1. The rep-rate computation must be done at low priority so that it does not starve out other IOC functions – particularly record processing and channel access.
2. The operator should have a visual indication that the rep-rate computation process is happening and when it is finished.
3. For variable rep-rates computed on the timing master IOC, the new patterns should not take effect until all the variable rep-rate gates have been re-computed. Once all the new patterns have been computed, they should all take effect immediately upon the next cycle.

Rep-rate pattern computation is done by EPICS record processing, since it is performed by gensub records. In order to comply with requirement 1, a low-priority scan task is created. This task, named

"scanRR", waits on a semaphore that is signaled when the "Load New Rep-Rate Values" button is pressed on the Timing Master EDM screen. Once the semaphore is taken, the task loops through the list of all variable rep-rate gates and calls "dbProcess" on their rep-rate gensub records. This ensures that all the rep-rate gensub records are processed, and that the processing occurs at a low enough priority that it will not interfere with regular record processing or with EPICS channel access.

This is also the reason that all rep-rate and constraint mux gensub records should have their SCAN fields set to "Passive".

If there are constraints, it is almost certain that one or more rep-rate gensub records will be processed two or more times. Each record will be processed once by the low-priority "scanRR" task. In addition, a record may be processed an additional time for each rep-rate or mux gensub record that links to it with a "Process Passive" link. In order to avoid redundant rep-rate pattern computations, a global sequence number is maintained by the "v124sSetNewRepRates" routine. This sequence number is incremented each time the "v124sSetNewRepRates" routine is called. When the rep-rate and constraint mux gensub routines compute a new pattern, they store the value of this sequence number in their rep-rate information structures along with the new pattern. When a rep-rate or constraint mux gensub record is processed, it compares the sequence number of its current pattern with the current value of the global sequence number. If the two match, the routine concludes that this is a redundant request and does not compute a new pattern.

Requirement 2 is satisfied by a global state variable maintained by the "v124sTriggerVariableGates" routine (see section 8.5.3 above). The variable, named "v124s_rr_status" has three states:

IDLE	=	No new rep-rate pattern computations have been performed recently.
SETTING_NEW_RATES	=	We are currently in the process of computing new rep-rate patterns. The previous patterns are still in effect.
NEW_RATES_SET	=	A new pattern computation completed within the last 10 seconds. The new patterns are currently in effect.

The state is set to "SETTING_NEW_RATES" by the "v124sSetNewRepRates" routine. When a rep-rate gensub record computes a new pattern, it sets a "new" flag in its rep-rate information structure. Once the "v124sTriggerVariableGates"routine (which is called once every cycle) sees that all the rep-rate gensub records have computed new patterns, it sets the state to "NEW_RATES_SET", clears all the "new" flags, and switches over to the new rep-rate patterns. The "NEW_RATES_SET" state remains for 10 seconds (one super-cycle) before reverting back to "IDLE".

Requirement 3 is met by keeping two copies of the rep-rate pattern. The rep-rate information structure contains an array, in list format, of every cycle for which the gate is supposed to fire. This is the most convenient format for the pattern computation process. In addition, the rep-rate gensub record maintains a 600 byte logical array which is TRUE (1) for each cycle that the gate should fire on and FALSE (0) for each cycle that the gate should not fire on. This is the most convenient format for producing rep-rate scope displays and for the "v124sTriggerVariableGates" routine, which determines whether or not to fire a variable rep-rate gate on the current cycle. When a new rep-rate pattern is computed, it is only stored in the rep-rate information structure – which is then available for use as a constraint pattern by dependent rep-rate gensub records. When the "v124sTriggerVariableGates" routine determines that all the rep-rate gensub records have computed new patterns, it re-computes the logical byte arrays from the rep-rate information structures.

The complete rep-rate computation process for the timing master IOC is described below. The process would be similar for a client IOC, but without the SNS event link transmissions.

- The operator sets new rep-rate values in the rep-rate setpoint records and presses the "Load New Rep-Rate Values" button on the Timing Master EDM screen.

- The “Load New Rep-Rate Values” button causes an event record to process. The device support for this event record is contained within the “repRateUtil.c” module. If the current state is “SETTING_NEW_REP_RATES”, the device support routine returns without taking any action. If we are not already setting rep-rates, the routine calls the “v124sSetNewRepRates” routine and then triggers EPICS event 251 (the “Compute-Rep-Rate” event). The “v124sSetNewRepRates” routine changes the state to “SETTING_NEW_REP_RATES”, increments the sequence number, and signals the semaphore that will wake up the “scanRR” task.
- A bo record on the timing master IOC is triggered by EPICS event 251 and broadcasts SNS event 251 (Compute-Rep-Rate) on the event link. This will let any client IOCs that are interested know that we are in the process of computing new rep-rates.
- The low-priority scan task, “scanRR”, wakes up on the semaphore given by the “v124sSetNewRepRates” routine. It loops through the linked list of all variable rep-rate gates and calls “dbProcess” on each of their rep-rate gensub records.
- If a rep-rate gensub record is dependent on another variable rep-rate gate, the gensub record support process routine will cause that record to be processed first by virtue of the fact that the link to its rep-rate information structure was marked “Process Passive”.
- When the “v124sRepRate” routine (the gensub record’s processing routine) is executed, it first checks the sequence number in its rep-rate information structure against the global sequence number set by the “v124sSetNewRepRates” routine. If the numbers are the same, the routine returns without computing a new rep-rate pattern. If the numbers are different, the routine will compute a new rep-rate pattern, store it in its rep-rate information structure along with the new sequence number, and set the “new” field of the rep-rate information structure to indicate that a new pattern has been computed.
- The “v124sTriggerVariableGates” routine, which is called once per cycle by the timing master sequencer task (or a similar high-priority task on the client IOCs), notices when all the variable rep-rate gates have new patterns. It then re-computes new pattern arrays for each of the variable rep-rate gates and resets their “new” flags. It then sets the state to “NEW_RATES_SET” and posts EPICS event 252 (“New-Rep-Rate”).
- Another bo record, similar to the one that broadcast the Compute-Rep-Rate event on the SNS event link, is triggered by EPICS event 252, and broadcasts SNS event 252 (New-Rep-Rate) on the SNS event link. This will let any client IOCs that are interested know that we have finished computing new rep-rates and that the new rep-rate patterns are now in effect.
- The ai records that contain the rep-rate readbacks are also triggered by EPICS event 252. These records read the rep-rate actually set from the VALA field of the rep-rate gensub records. Note that the readback may be less than the setpoint if there were limiting dependencies.
- 600 cycles later (10 seconds), the “v124sTriggerVariableGates” routine sets the state back to “IDLE”.

The rep-rate pattern computation process is summarized by the sequence diagram shown below:

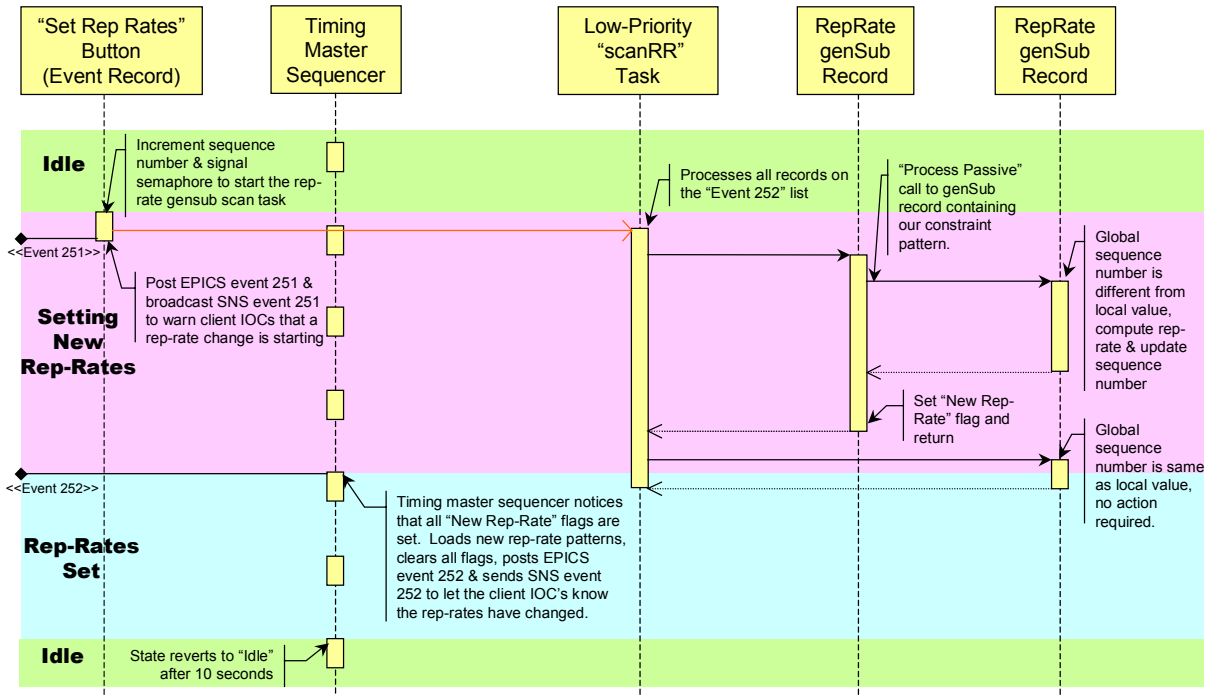


Figure 42
Sequence Diagram For Setting New Rep-Rate Patterns

The EPICS database for a variable rep-rate gate will require a rep-rate gensub record, an ao record for the rep-rate setpoint, an ai record for the rep-rate readback, and an event record to start the rep-rate computation process. Since the event record will cause all variable rep-rates to be re-computed, only one event record is required per IOC.

Figure 43 below shows a sample database for a variable rep-rate gate – including a constraint record and the event record that starts the whole process.

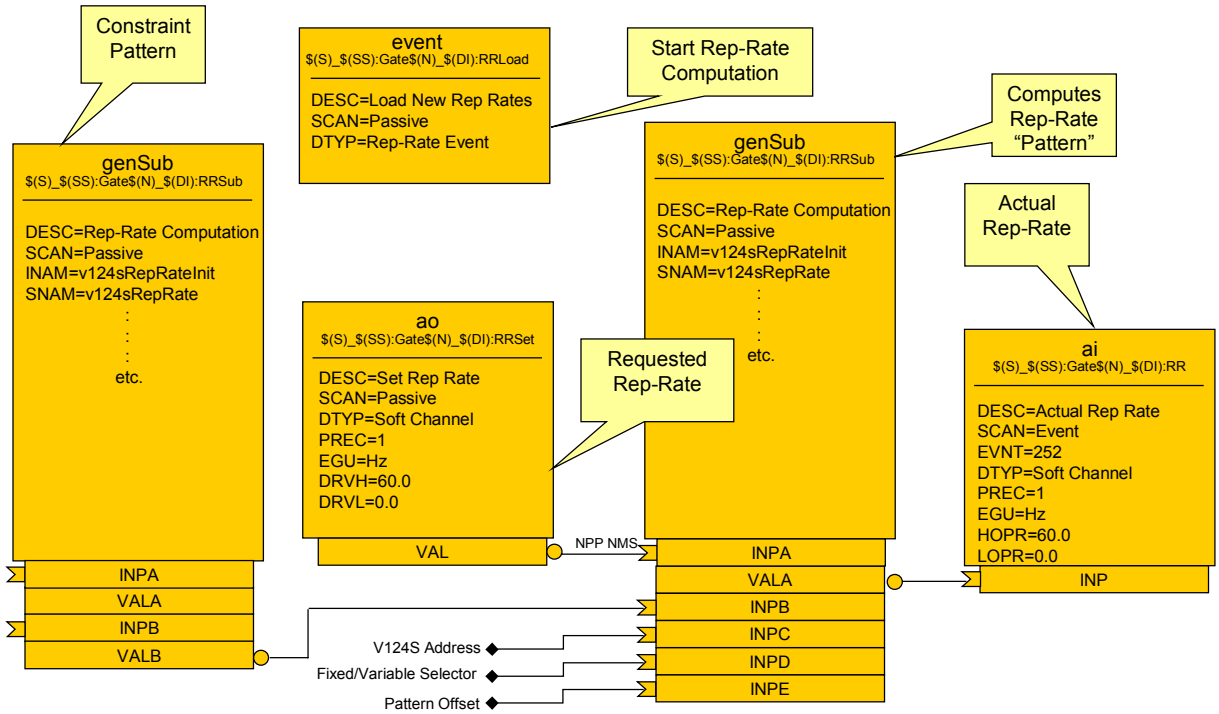


Figure 43
Sample Database for Variable Rep-Rate Gate

Note that the gensub record's SCAN field is "Passive" and that the ao record for the requested rep-rate does not forward link to it. We do not want to use normal EPICS record processing tasks (or priorities) to process the variable rep-rate gensub records. The ai record for the actual rep-rate has its SCAN field set to "Event", and the EVNT field set to 252. This will cause it to process after all the variable rep-rate patterns have been recomputed and the new rep-rate patterns are in effect.

The file, "variableRepRate.template" in the "$SHARE/timing/production/db" directory provides a template for creating variable rep-rate gates (see section 8.3).

8.6 TIME CONVERSIONS

Converting between timing system units (turns, sub-revolutions) and "real time" (microseconds) requires knowing what the ring revolution period is. This value is measured with a Joerger scalar module as described in section 7 and transmitted on RTDL frame 4. Time conversion gensub records obtain this value from the SNS utility module.

The software for implementing time conversion resides in the file:

```
$SHARE/timing/production/timingApp/src/timingSubs.c
```

This file is built into the "snsTimingLib" library which should be loaded by any IOC that uses a V124S card.

8.6.1 Turns To Microseconds

The gensub record for converting timing system units into microseconds has three inputs (turns, sub-revolutions, and fine delay) and one output (microseconds). It is used mostly to convert gate widths into microseconds, but could also be used for converting gate delays – provided that if the number of turns is greater than zero, the gate is triggered at the start of a revolution.

Not all the inputs need to be present. For example, if you are using this record to compute the actual width of gate, the gate's raw width would be input into the "sub-revolution" input and the "turns" and "fine delay" inputs would be zero.

This record should be set up as follows:

Subroutine Names:

INAM = v124sTurns2usecInit (Name of Initialization Routine)
SNAM = v124sTurns2usec (Name of Processing Routine)

Input Links:

INPA = Number of turns in the delay or width (integer value).

INPB = Number of sub-revolutions in the delay or width (integer value).

INPC = Fine delay (floating point value).

Output Links:

OUTA = The delay or width value expressed in microseconds (floating point value).

Field Type Declarations:

FTA	= LONG	(Field type for Input A – Turns)
FTB	= LONG	(Field type for Input B – Sub-Revolutions)
FTC	= DOUBLE	(Field type for Input C – Fine Delay)
FTVA	= DOUBLE	(Field type for Output A)

The initialization routine specified in the INAM field (v124sTurns2usecInit) opens a handle to the SNS utility module and registers its interest in RTDL frame 4. If the utility module is present and functioning, the value found in RTDL frame 4 is used for the conversion constant. If we cannot read the

measured ring revolution period, the record will use the nominal 1 GeV value (945.38776 nanoseconds) so that the converted times are at least reasonable.

8.6.2 Microseconds To Turns

The gensub record for converting real time (microseconds) to turns has one input and three outputs (turns, sub-revolutions, fine delay). It is somewhat less useful than the "turns to microseconds" record, since if the conversion value changes, the gate values will have to be re-set.

As before, the assumption for setting a gate delay is that the gate is triggered at the start of a revolution. Otherwise the delay calculation will probably be too long.

If you want to use this record to convert a gate width from microseconds to sub-revolutions, you will need to multiply the "turns" output by 32, add it to the "sub-revolutions" output, and then use the "fine delay" output for rounding up or down.

This record should be set up as follows:

Subroutine Names:

INAM = v124sUsec2turnsInit (Name of Initialization Routine)
SNAM = v124sUsec2turns (Name of Processing Routine)

Input Links:

INPA = Microsecond value to be converted (floating point value).

Output Links:

OUTA = Number of turns in the input value (integer value).

OUTB = Number of sub-revolutions in the input value (integer value).

OUTC = Fine delay (floating point value).

Field Type Declarations:

FTA	= DOUBLE	(Field type for Input A)
FTVA	= LONG	(Field type for Output A – Turns)
FTVB	= LONG	(Field type for Output B – Sub-Revolutions)
FTVC	= DOUBLE	(Field type for Output C – Fine Delay)

The initialization routine specified in the INAM field (v124sUsec2turnsInit) performs exactly the same functions as the initialization routine for the "turns to microseconds" record (v124sTurns2usecInit). In fact, the only thing the v124sUsec2turnsInit does is call v124sTurns2usecInit.

9 UTILITY MODULE

9.1 HARDWARE

T.B.S.

9.1.1 Jumper Settings

Two addresses need to be set on the utility module – the VME A24 base address for the card, and the card's IOC Reset ID. Figure 44 (below) shows the locations of these jumpers.

The VME A24 base address is set via ten jumpers located in the upper middle section of the card. The jumpers set base address lines A14 - A23 (16K boundary). With the board standing upright (as shown below), the high-order bit is at the top and the low order bit is toward the bottom. If a jumper is present, it represents a logical "0". If the jumper is removed, it represents a logical "1".

The IOC Reset ID is set by 24 jumpers located toward the front of the card. If the utility module sees an value in the "IOC Reset" RTDL frame (frame 15) that matches the value set in these jumpers, it will assert the VME SYSRESET line, causing the IOC to reboot. As with the VME address jumpers, the high-order bit is toward the top of the card and the low-order bit is at the bottom. If a jumper is present, it represents a logical "0". If the jumper is removed, it represents a logical "1".

The SNS standard is to set the IOC Reset ID to the last three octets of the IOC's IP address.

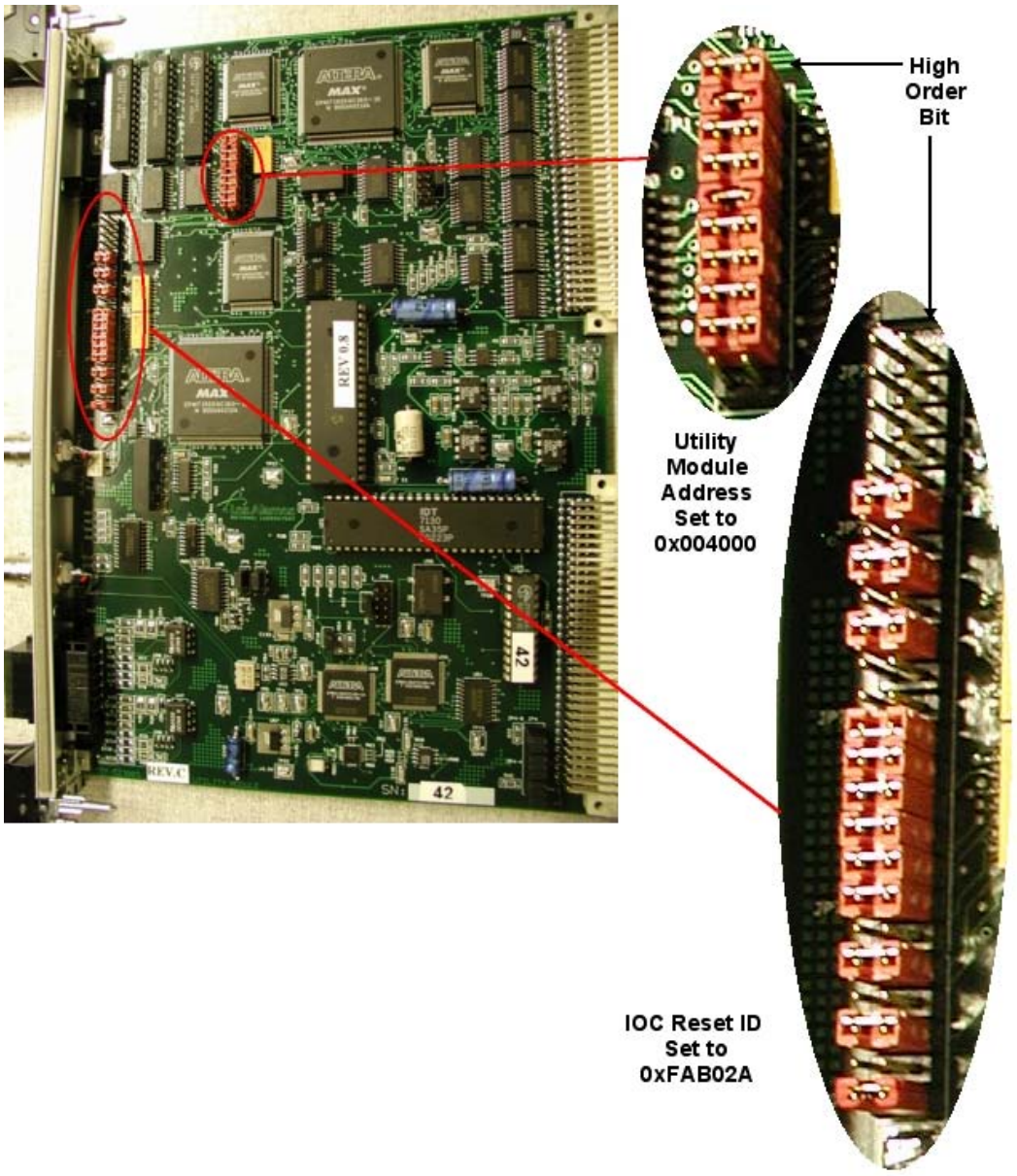


Figure 44
Jumper Settings for Utility Module

9.2 DIAGNOSTIC SCREENS

The main diagnostic screen for the SNS utility module is in
`$SHARE/utility/production/opi/utilStatus.edl`

A typical utility module diagnostic screen is shown below:

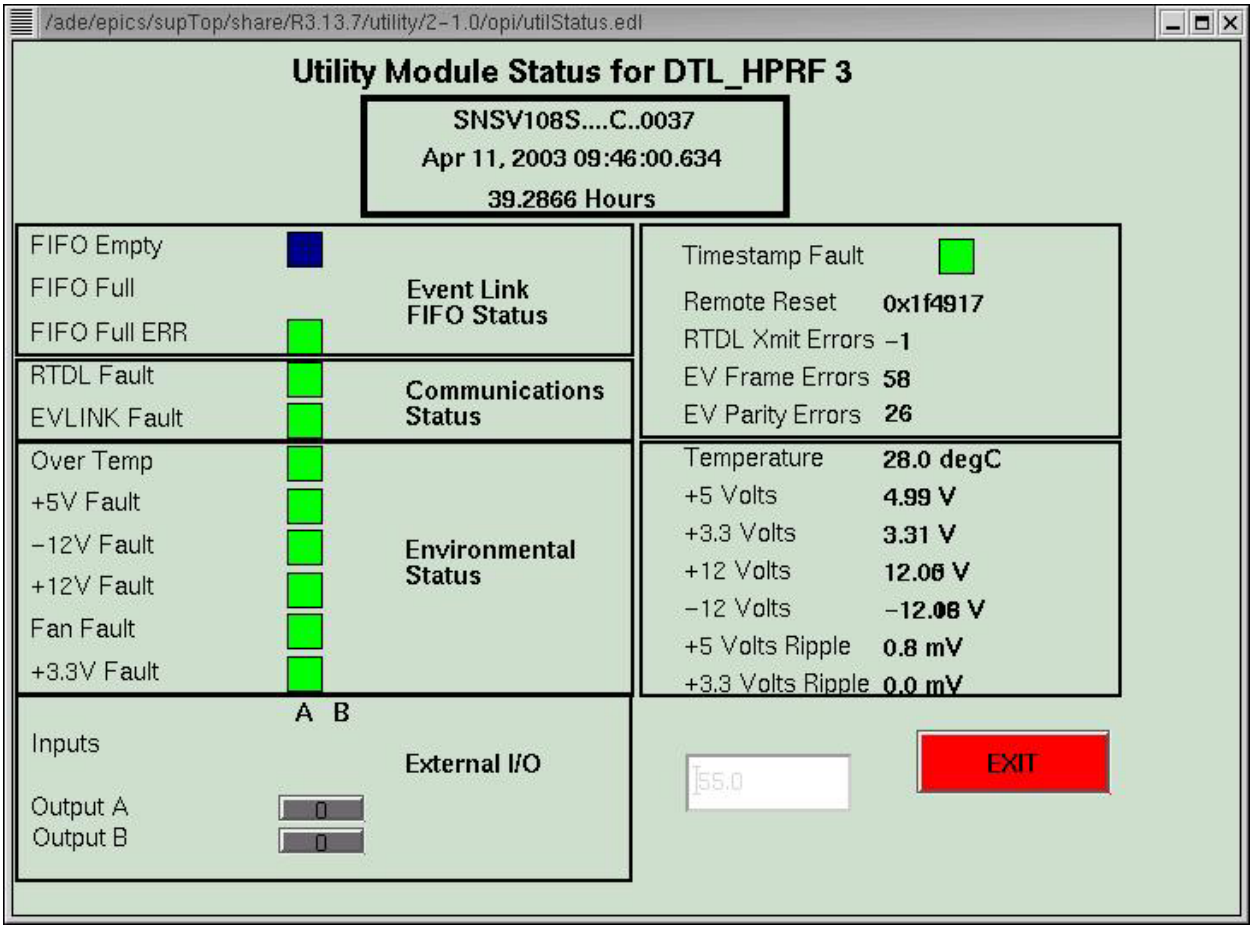


Figure 45
Utility Module Diagnostic Screen

The box at the top of the screen displays the utility module's ID, hardware revision level, and serial number. It also displays the current time (as read from the RTDL) and number of hours the IOC has been running. The boxes on the left half of the screen show the environmental and system status indications. The boxes on the right half of the screen show the actual voltage, temperature and ripple levels. The over-temperature trip point can be set in the white box on the lower right.

The next figure shows a utility module screen for an IOC in a VXI crate. Notice that there is no 3.3 volt measurement in the VXI crate. There is also no ripple measurement.

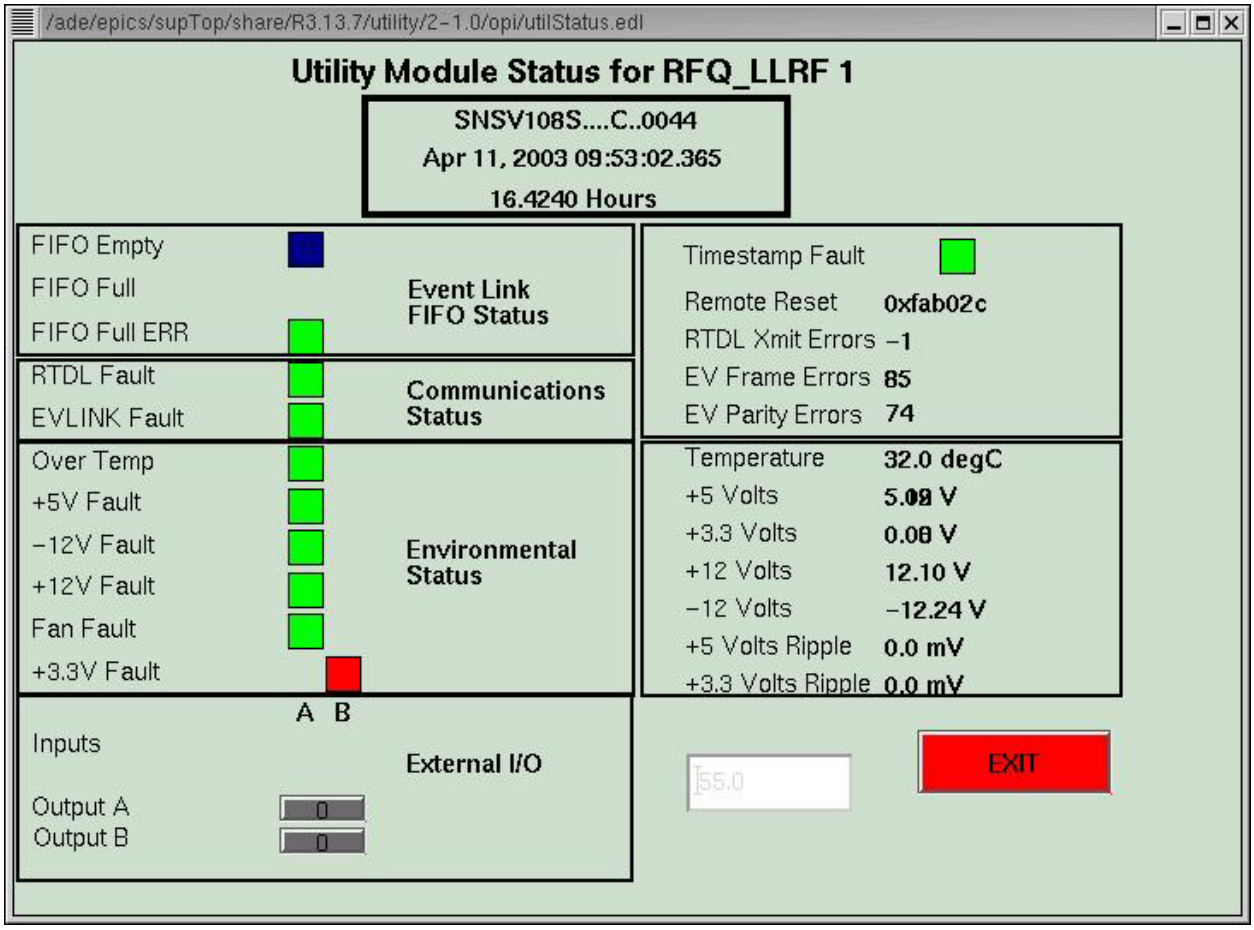


Figure 46
Utility Module in a VXI Crate

9.3 EPICS SOFTWARE

The EPICS software for the utility module resides in the directory:

```
$SHARE/utility/production/
```

The "bin", "db", "dbd", "include", and "opi" directories all contain useful items. The "bin" directory contains architecture-specific sub-directories, each of which contain the file:

```
snsUtilLib
```

This file contains the EPICS driver and device support for the utility module. It also contains EPICS timestamping routines which allow EPICS records to be timestamped with the Cycle-Start time read from the RTDL.

The "include" directory contains two header files. "drvSnsUtil.h" contains the definitions and function templates needed to call utility module driver routines. "snsTiming.h" defines the SNS event and RTDL frame numbers. It is included here, rather than in the timing directory, because the utility module software needs to know these values. Thus the utility module software needs to be built before the timing system software (see section 8.3.1).

The "dbd" directory contains the following .dbd files:

- **snsUtilInclude.dbd:** This is an unexpanded dbd include file that references all the record types and all the driver and device definitions needed by the database templates in the "db" directory. This file is suitable for inclusion in an application's "*xxx*AppInclude.dbd" file.
- **snsUtilApp.dbd:** This file is a fully expanded version of "snsUtilInclude.dbd". This file is primarily used by vdct as the reference dbd file when constructing or editing the template files.

The "db" directory contains template files for displaying the basic environment and status outputs monitored by the utility module. The two template files that should be included with each IOC are:

snsUtilEnviron.db

This database template contains the records for monitoring the crate temperature, voltages and ripple. It also contains an ao record which will allow you to set the crate "Overtemperature" trip point.

Macros:

S = System Name
SS = Sub-System Name
N = System Instance
DI = Device Instance

The above macros are used to construct the PV names, which are of the form:

$(S)_$(SS):UTIL$(N)_$(DI):<signal>

Typically, the PV names follow the name of the IOC:

snsUtilStatus.db

This database template contains the records for monitoring the status and fault information. It also contains records with information about the module ID, timestamp, and IOC Reset ID.

Macros:

S = System Name
SS = Sub-System Name
N = System Instance
DI = Device Instance

The above macros are used to construct the PV names, which are of the form:

$(S)_$(SS):UTIL$(N)_$(DI):<signal>

Typically, the PV names follow the name of the IOC:

9.3.1 Setting Up a Utility Module Client

9.3.1.1 config Directory RELEASE File

The first step in adding a V124S card to your IOC is to go to the top level "config" directory, and edit the RELEASE file. Make sure the following symbol (or its equivalent) is defined:

```
UTILITY = $(SHARE_RELEASE)/utility/production
```

This will make sure that the other application Makefiles have access to all the utility module definitions and code they need.

9.3.1.2 src Directory Makefile.Vx File

The SNS standard is for an application to copy all the system and utility software it needs into its local "bin" directory and load it from the local directory at boot time. If your IOC implements this method, you will need to edit the "Makefile.Vx" file in your application's "src" directory and add the following line:

```
BIN_INSTALLS += $(UTILITY_BIN)/snsUtilLib
```

This line will cause a "make" in the "src" directory to copy the latest versions of the utility module software into your local "bin" directory.

9.3.1.3 src Directory *xxx*AppInclude.dbd File

A good practice is to create an application-wide ".dbd" file. The utility module dbd directory contains an "snsUtilInclude.dbd" file that contains all the record, device, and driver definitions needed to support the utility module databases. Any duplicate definitions will get sorted out when the *xxx*AppInclude.dbd file is expanded into the *xxx*App.dbd file during the make process (note that duplicate .dbd definitions are not sorted out when several .dbd files are loaded at boot time – this is why the *xxx*AppInclude.dbd method is preferred).

To make sure all the relevant utility module definitions are included in your application's .dbd file, insert the following line into your *xxx*AppInclude.dbd file:

```
include "snsUtilInclude.dbd"
```

9.3.1.4 src Directory base.dbd and basLIBOBJS Files

If you are using the *xxx*AppInclude.dbd method to build an application-wide .dbd file, then the only thing you need to worry about including in the "base.dbd" file are the base record, device, and driver support routines that your own application uses.

The baseLIBOJBS file, on the other hand should be edited to include all record types used by the utility module databases. These record types are:

```
aiRecord        aoRecord        biRecord        boRecord
calcRecord      eventRecord     longinRecord    subRecord
stringinRecord
```

9.3.1.5 Db Directory *xxx*App.substitutions File

Each IOC with a utility module card should include the basic utility module environment and status databases.

If you use a substitutions file to create your databases, you can define the utility module database here with lines of the form:

```
#--------------------
# Utility Module Database
#
file snsUtilEnviron.db {
    {S=<<system>> SS=<<sub-system>>, N=<<system-instance>>, DI=<<device-instance>>}
}

file snsUtilStatus.db {
    {S=<<system>> SS=<<sub-system>>, N=<<system-instance>>, DI=<<device-instance>>}
}
```

The system, sub-system, system instance, and device instance names are normally derived from the IOC name.

9.3.1.6 iocBoot Directory st.cmd File

The IOC startup command file will require several modifications. First, the utility module support library must be loaded. If you load all your software from your local "bin" directory, then the following lines should be added to your st.cmd file:

```
cd appbin
   :
<< Other System Libraries >>
   :
ld < snsUtilLib
```

Next you need to initialize the utility module hardware and configure any interrupts you wish for it to take. At the minimum, you should configure the event interrupt so that the utility module can do timestamping and monitor the RTDL frames. Before you call "iocInit", you should have the following lines:

```
#-------------------------------------------------
# Initialize the Utility Module
#-------------------------------------------------

snsUtilInit 0, 0x4000
snsUtilConfigEvent 0, 0x40, 5
```

9.4 DRIVER SOFTWARE

The EPICS driver and device support software for the utility module reside in the files:

```
$SHARE/utility/production/utilityApp/src/drvSnsUtil.c
$SHARE/utility/production/utilityApp/src/devSnsUtil.c
```

These files are built into the "snsUtilLib" library which needs to be loaded by any IOC that uses a utility module. The header file "drvSnsUtil.h" (in the same directory) contains the interface definitions and is installed in the directory:

```
$SHARE/utility/production/include
```

The routines below are implemented in the drvSnsUtil.c module.

SNSUTIL = snsUtilInit (card, base)

This routine is called from the vxWorks startup file to configure the utility module's VME address. It returns a handle to the utility module card (in case you want to call other utility module driver routines from the vxWorks shell).

Arguments:

card = Card number (usually 0).
base = Base address in A24 space.

Returns:

Utility module handle if the utility module was located and correctly configured. NULL if there was an error.

status = snsUtilConfigEvent (card, vector, level)

This routine is called from the vxWorks startup file to configure the utility module's event interrupt vector and level. Event interrupts must be enabled before the utility module can do timestamping or RTDL frame monitoring

Arguments:

card = Card number (usually 0).
vector = Interrupt vector for the utility module's event interrupt.
level = Interrupt request level for the utility module's event interrupt.

Returns:

OK = Utility module's event interrupt was successfully configured.
ERROR = Utility module's event interrupt configuration had errors.

status = snsUtilConfigEnviron (card, vector, level)

This routine is called from the vxWorks startup file to configure the utility module's environment interrupt vector and level. Environment interrupts allow you to process EPICS records (via I/O Int) whenever a power supply, overtemperature, event link, or RTDL link error is encountered.

Arguments:

card	=	Card number (usually 0).
vector	=	Interrupt vector for the utility module's environment interrupt.
level	=	Interrupt request level for the utility module's environment interrupt.

Returns:

OK	=	Utility module's environment interrupt was successfully configured.
ERROR	=	Utility module's environment interrupt configuration had errors.

status = snsUtilConfigExtern (card, vector, level)

This routine is called from the vxWorks startup file to configure the utility module's external I/O interrupt vector and level. External I/O interrupts allow you to process EPICS records (via I/O Int) whenever one of the two external input lines changes state.

Arguments:

card	=	Card number (usually 0).
vector	=	Interrupt vector for the utility module's external I/O interrupt.
level	=	Interrupt request level for the utility module's external I/O interrupt.

Returns:

OK	=	Utility module's external I/O interrupt was successfully configured.
ERROR	=	Utility module's external I/O interrupt configuration had errors.

10 TIMING SYSTEM SIGNAL NAMES

10.1 DEVICE NAMES

The following eight devices are defined by the SNS timing system:

Gate	Signals associated with individual timing gate triggers.
GateGen	Global signals for the timing gate trigger module (V124S).
Event	Signals associated with individual events generated on the SNS event link.
EventGen	Global signals for the SNS event link encoder module (V123S)
RTDL	Signals associated with a single RTDL frame generated by the SNS RTDL encoder.
RTDLIn	Global signals associated with the RTDL input module (V206)
RTDLGen	Global signals associated with the RTDL encoder module (V105)
UTIL	Signals associated with the SNS Utility Module

10.2 TIMING GATE TRIGGER MODULE GLOBAL SIGNALS

The timing gate trigger module (V124S) has several signals that are global to the entire card. The record names follow the template:

$(S)_$(SS):GateGen$(N)_$(DI):<*signal*>

where:

S = System Name
SS = Subsystem Name
N = System Unit Number
DI = Device Instance (typically the V124S card letter)

A typical record name would be:

MEBT_Tim:GateGen_A:OffsetDly

The EPICS records for timing gate trigger signals should have DTYP="Timing Gate". The INP/OUT field should be of the form:

#C*x* **S0 @**<*function*>

Where "*x*" is the card number (starting at 0), and <*function*> is a brief text string parameter indicating what the record does.

The following table lists the standard signal names for global timing gate trigger records, along with their record types, and function parameter values:

Timing Gate Trigger Module (V124S) Global Signals			
Signal Name	**Record Type**	**Function Parameter**	**Description**
CarrierError	longin	@Carrier Error	Event Link carrier error counter
CarrierStat	bi	@Carrier Error	Event Link carrier error status bit
ClearErrors	bo	@Clear Error	Clears all error counters
Clock	bi	@Clock	Read source of card's RF clock (internal or external)
ClockSet	bo	@Clock	Sets source of card's RF clock (internal or external)
Enable	bi	@Enable	Indicates if card is enabled

Timing Gate Trigger Module (V124S) Global Signals			
Signal Name	**Record Type**	**Function Parameter**	**Description**
EnableReg	longin	@Enable Reg	Contents of card's enable register (on-line & clock enable)
EnableSet	bo	@Enable	Enables card for processing
EventLinkStat	bi	@EV Link Error	Event link overall error summary status bit
FrameError	longin	@Frame Error	Event link frame error counter
FrameStat	bi	@Frame Error	Event link frame error status bit
HaltStat	longin	@Halt Status	Indicates which channels are halted
IntEna	longin	@Int Enable	Indicates what interrupt conditions are enabled for this card.
LastCleared	stringin	@TS Clear	Last time the card's error counters were cleared.
LinkStat	longin	@Link Status	Contents of card's event link status register (carrier, frame, parity)
OffsetDly	ai	@Offset Delay	Offset delay for the entire V124S card
OffsetDlySet	ao	@Offset Delay	Offset delay setpoint for the entire V124S card
ParityError	longin	@Parity Error	Event link parity error counter
ParityStat	bi	@Parity Error	Event link parity error status
RFSync	longin	@RF Sync	Read the revolution frequency resynch event number
RFSyncSet	longout	@RF Sync	Set the revolution frequency resynch event number
TStat	longin	@TS Status	Indicates which channels have set timestamp values
TSync	longin	@TS Sync	Read the timestamp reset event number
TSyncSet	longout	@TS Sync	Set the timestamp reset event number

10.3 TIMING GATE TRIGGER CHANNEL SIGNALS

The record names for signals belonging to individual timing gates follow the template:

$(S)_$(SS):Gate$(N)_$(DI):<*signal*>

where:

S = System Name
SS = Subsystem Name
N = System Unit Number
DI = Device Instance (typically the name of the device controlled by the gate)

A typical record name would be:

MEBT_Diag:Gate_BPM05:Width

Unless they are soft channels, the EPICS records for timing gate trigger signals should have DTYP="Timing Gate". The INP/OUT field should be of the form:

#C*x* **S***n* **@**<*function*>

Where "*x*" is the card number (starting at 0), "*n*" is the gate number (starting at 1), and <*function*> is a brief text string parameter indicating what the record does.

The following table lists the standard signal names for individual timing gate trigger records, along with their record types, and function parameter values. Not all of these signals may be present for any given gate.

Timing Gate Trigger Module (V124S)Individual Gate Signals			
Signal Name	**Record Type**	**Function Parameter**	**Description**
AutoReload	bi	@Auto Reload	Read channel's "Auto Reload Counters" bit
AutoReloadSet	bo	@Auto Reload	Set channel's "Auto Reload Counters" bit

Timing Gate Trigger Module (V124S)Individual Gate Signals			
Signal Name	**Record Type**	**Function Parameter**	**Description**
ChanCntrl	longin	@Chan Ctrl	Contents of the channel control register (16 bits)
CounterStat	longin	@Counter Stat	Contents of the counter status register (5 bits)
Event	longin	@Event	Read the event to trigger the channel on (0 = no event)
EventSet	longout	@Event	Set the event to trigger the channel on (0 = no event)
FineDly	ai	@Fine Delay	Channel's fine delay value (nanoseconds)
FineDlySet	ao	@Fine Delay	Set channel's fine delay (nanoseconds)
HaltDlyEna	mbbi	@Halt Delay	Halt delay counter trigger selection (Fine Delay, No Halt, Next Chan)
HaltDlyEnaSet	mbbo	@Halt Delay	Select the halt delay counter trigger (Fine Delay, No Halt, Next Chan)
ManualTrigger	bo	@Trigger	Trigger channel from VME command
Polarity	bi	@Polarity	Output polarity value
PolaritySet	bo	@Polarity	Set output polarity
Reset	bo	@Reset	Reset channel to a predefined (and off) state
RevDly	longin	@Rev Delay	Revolution delay value
RevDlySet	longout	@Rev Delay	Set revolution delay value
RevDlyLSB	longin	@Rev LSB	Least significant byte of the current value of the revolution delay counter
RevDlyEna	mbbi	@Rev Delay	Revolution counter trigger selection (Manual, Event, External, Prev Chan)
RevDlyEnaSet	mbbo	@Rev Delay	Select the revolution counter trigger (Manual, Event, External, Prev Chan)
RR	ai	(soft channel)	Actual rep-rate
RRSet	ao	(soft channel)	Requested rep-rate
RRSub	gensub	(soft channel)	Rep-Rate pattern computation record
SingleShot	bo	@SS Arm	Arm channel for single shot
Stop	bi	@Stop	Read channel's "Stop Counters" bit
StopSet	bo	@Stop	Set channel's "Stop Counters" bit
SubRevDly	longin	@SubRev Delay	Sub-revolution delay value
SubRevDlySet	longout	@SubRev Delay	Set sub-revolution delay
SubRevDlyLSB	longin	@SubRev LSB	Least significant byte of the current value of the sub-revolution delay counter
SubRevDlyEna	mbbi	@SubRev Delay	Sub-revolution counter trigger selection Manual, Event, Rev Start, Rev Done)
SubRevDlyEnaSet	mbbo	@SubRev Delay	Select the sub-revolution counter trigger (Manual, Event, Rev Start, Rev Done)
Timestamp	longin	@Timestamp	Read channel's timestamp value
TriggerCnt	longin	@Trigger Count	Number of gates to generate after receiving a trigger
TriggerCntSet	longout	@Trigger Count	Set the number of gates to generate after receiving a trigger
TSClock	bi	@TS Clock	Channel's timestamp clock source (carrier clock or event)
TSClockSet	bo	@TS Clock	Set channel's timstamp clock source (carrier clock or event)
TSClockEvent	longin	@TS Clock	Event to use as the timestamp clock (if event counting is enabled)
TSClockEventSet	longout	@TS Clock	Set the event used as the timestamp clock (if event counting is enabled)
TSConfig	longin	@TS Config	Value of channel's timestamp configuration register (4 bits)
TSTrigEvent	longin	@TS Trig	Event to use to trigger timestamping (if event triggering is enabled)
TSTrigEventSet	longout	@TS Trig	Set the event used to trigger timestamping (if event triggering is enabled)
TSTrigger	mbbi	@TS Trig	Timestamp trigger selection (None, Event, 1st Pulse, Last Pulse)
TSTriggerSet	mbbo	@TS Trig	Select timestamp trigger (None, Event, 1st Pulse, Last Pulse)

Timing Gate Trigger Module (V124S)Individual Gate Signals			
Signal Name	**Record Type**	**Function Parameter**	**Description**
uSecDly	gensub	(soft channel)	Computed gate delay in microseconds
uSecWidth	gensub	(soft channel)	Computed gate width in microseconds
Width	longin	@Width	Gate width (sub-revolutions)
WidthSet	longout	@Width	Set gate width (sub-revolutions)

10.4 EVENT LINK ENCODER GLOBAL SIGNALS

The event link encoder module (V123S) has several signals that are global to the entire event link system. These record names follow the template:

$(S)_Tim:EventGen$(N):<*signal*>

where:

S = System Name

N = System Unit Number (blank for the timing master, 2 for the backup)

A typical record name would be:

ICS_Tim:EventGen:OnLine

The EPICS records for event encoder signals should have DTYP="Event Link". The INP/OUT field should be of the form:

#C0 S0 @<*function*>

Since there is only one event master module, the card and signal numbers are always 0. The <*function*> parameter is a brief text string indicating what the record does.

The following table lists the standard signal names for global event link records, along with their record types, and function parameter values:

Event Link Encoder Module (V123S) Global Signals			
Signal Name	**Record Type**	**Function Parameter**	**Description**
ClearErrors	bo	@Clear Error	Clear all error counters
Clock	mbbi	@Clock	V123S Clock source (Internal, External, or Auto)
ClockInUse	bi	@Clock In Use	Which clock (internal or external) is currently being used to generate the event link.
ClockSet	mbbo	@Clock	Set V123S clock source (Internal, External, or Auto)
CommCnt	longin	@Comm Err	Number of times a communication error occurred between the V123S and a V101S module.
CSReg	longin	@CSR Reg	Contents of V123S Command/Status register (bitmask)
DBLPPCnt	longin	@Dbl PP Err	Number of Pre-Pulse double trigger errors seen.
DblT0Cnt	longin	@Dbl T0 Err	Number of T0 double trigger errors seen.
DblTextCnt	longin	@Dbl Text Err	Number of Text double trigger errors seen.
EncInt	bi	@Encoder Int	State of V123S interrupt enable bit
EncIntSet	bo	@Encoder Int	Enable or disable interrupts on the V123S module
ErrorReg	longin	@Error Reg	Last value of V123S Error Status register (bitmask)
FIFOFullCnt	longin	@FIFO Full Err	Number of times the soft event FIFO was full and couldn't take another event.
FIFOStat	mbbi	@FIFO Stat	Status of soft event FIFO (Empty, Active, Full)
HardRangeCnt	longin	@Hard Range Err	Number of times system tried to generate a hardware event outside the range of valid hardware events
LastCleared	stringin	@TS Clear	Timestamp of last time error counters were cleared.
MapWriteEna	bo	@Map Write Ena	Enable/Disable the event map table for writes
OnLine	bi	@On Line	On-Line status of V123S module

Event Link Encoder Module (V123S) Global Signals			
Signal Name	**Record Type**	**Function Parameter**	**Description**
OnLineSet	bo	@On Line	Set V123S module on or off line.
RFClockBadCnt	longin	@RF Clock Err	Number of failures detected on the RF-Clock input.
RFClockStat	bi	@RF Clock Stat	Status of the external RF clock signal.
SoftRangeCnt	longin	@Soft Range Err	Number of times system tried to send a software event with a hardware event number
SoftTrig	longout	@Soft Trig	Trigger soft event
StatusReg	longin	@Status Reg	Contents of Real-Time Status register (bitmask)
TimingErrCnt	longin	@Timing Err	Number of collisions seen between Pre-Pulse, T0, or Text and other events.
VMECnt	longin	@VME Err	Number of VME access errors seen.

10.5 SIGNALS FOR INDIVIDUAL EVENTS

Each event that goes out on the SNS event link has several signals associated with it. Most of the signal names in this section only refer to hardware events – that is, events generated by the V123S and V101S modules. The one exception to this is the "Trigger" record, which can trigger both hardware and software events. The record names for individual events follow the template:

$(S)_Tim:Event$(N)_$(EVENT):<*signal*>

where:

S = System Name
N = System Unit Number (blank for the timing master, 2 for the backup)
EVENT = Event name or number. Typically this is the "V101 number" of the event, based on the V101 channel that generates it.

A typical record name would be:

ICS_Tim:Event_6:Trigger

The EPICS records for event signals should have DTYP="Event Link". The INP/OUT field should be of the form:

#C0 S<*event*> **@**<*function*>

Where <*event*> is the V101 number (based on the V101 channel that generates it) of the event we wish to control. The <*function*> parameter is a brief text string indicating what the record does.

The following table lists the standard signal names for event records, along with their record types, and function parameter values:

Individual Event Signals			
Signal Name	**Record Type**	**Function Parameter**	**Description**
Ena	bi	@Enable	Indicates whether the event is enabled for broadcast or not.
EnaSet	bo	@Enable	Enables or disables the event for broadcast on the event link.
LostCnt	longin	@Lost Event Err	Number of times an event trigger did not generate an event (usually due to double-triggering).
Map	longin	@Map	Mapping between hardware event number and the event number actually broadcast on the event link.
MapSet	longout	@Map	Set the mapping between the hardware event and the event number actually broadcast on the event link.
Trigger	bo	@Trigger	Trigger the specified event. Note: Unlike the other records listed here, the signal parameter in the OUT field can specify either a hardware or a software event.

10.6 RTDL ENCODER GLOBAL SIGNALS

The RTDL encoder module (V105) has nine signals that are global to the whole card. The record names follow the template:

$(S)_Tim:RTDLEncoder$(N):<*signal*>

where:

S = System Name (Always ICS in the accelerator and Dev for lab systems)
N = System Unit Number (blank for the timing master, 2 for the backup)

A typical record name would be:

ICS_Tim:RTDLEncoder:Enable

The EPICS records for RTDL encoder card signals should have DTYP=" RTDL V105 Encoder". The INP/OUT field should be of the form:

#C0 S*x*

Where "*x*" is the signal number of the V105 device support signal (starting from 0). The card number is always 0 since a timing system can only have one RTDL encoder.

The following table lists the standard signal names for global RTDL encoder module records, along with their record types, and VMEIO signal number values:

RTDL Encoder (V105) Global Signals			
Signal Name	**Record Type**	**VME Signal**	**Description**
NoReplyError	longin	#C0 S0	Count of no response errors from input module channels
NoReplyErrorID	longin	#C0 S1	ID code for the last no reply error
NoCarrierError	longin	#C0 S2	Count of no carrier errors
Status	longin	#C0 S3	
TriggerSource	bi	#C0 S0	Reads the trigger source which controls sending the RTDL frame, can be Event or External
TriggerSourceSet	bo	#C0 S0	Sets the trigger source which controls sending the RTDL frame, can be Event or External
CardReset	bi	#C0 S2	The state of the card reset bit
CardResetSet	bo	#C0 S2	Sets or resets the card reset bit
ClearErrors	bo	#C0 S1	Clears the driver's error counters

10.7 RTDL INPUT MODULE GLOBAL SIGNALS

The RTDL input module (V206) as four signals that are global to the whole card. The record names follow the template:

$(S)_Tim:RTDLIn$(N)_$(DI):<*signal*>

where:

S = System Name
N = System Unit Number (blank for the timing master, 2 for the backup)
DI = Device Instance (typically the V206 card designator (A,B,C,…))

A typical record name would be:

ICS_Tim:RTDLIn_A:Enable

The EPICS records for RTDL input card signals should have DTYP="RTDL Input". The INP/OUT field should be of the form:

#C*x* **S0 @**<*function*>

Where "*x*" is the card number of the V206 card (starting from 0). The <*function*> parameter is a brief text string indicating what the record does.

The following table lists the standard signal names for global RTDL input module records, along with their record types, and function parameter values:

RTDL Input Module (V206) Global Signals			
Signal Name	**Record Type**	**Function Parameter**	**Description**
ClearErrors	bo	@Clear Error	Clears all error counters for this card.
Enable	bi	@On Line	Whether the V206 card is enabled to generate RTDL frames
EnableSet	bo	@On Line	Enable/Disable V206 card from generating RTDL frames.
LastCleared	stringin	@TS Clear	Last time the V206 error counters were cleared

10.8 SIGNALS FOR RTDL INPUT CHANNELS

Each input channel on the RTDL input module (V206) has several signals associated with it. The record names for individual input channels follow the template:

$(S)_Tim:RTDL$(N)_(DI)(CH):<*signal*>

where:

S = System Name
N = System Unit Number (blank for the timing master, 2 for the backup)
DI = Device Instance (the V206 card designator (A,B,C,…))
CH = Channel Number (starting at 1)

A typical record name would be:

ICS_Tim:RTDL_A1:Enable

The EPICS records for RTDL input channel signals should have DTYP="RTDL Input". The INP/OUT field should be of the form:

#C*x* **S**$(CH) **@**<*function*>

Where "*x*" is the card number of the V206 card (starting from 0) and "$(CH)" is the channel number (starting from 1 – as above). The <*function*> parameter is a brief text string indicating what the record does.

The following table lists the standard signal names for global RTDL input module records, along with their record types, and function parameter values:

RTDL Input Module (V206) Signals for Individual Channels			
Signal Name	**Record Type**	**Function Parameter**	**Description**
CRCError	longin	@CRC Error	Count of CRC errors on the input link (only used when the data source is "External")
Data	longin	@Frame Data	Reads the RTDL data word for this channel
DataSet	longout	@Frame Data	Sets the RTDL data word for this channel
Enable	bi	@Enable	Enable/Disable status of the channel
EnableSet	bo	@Enable	Enables or Disables the channel for RTDL output
FrameError	longin	@Frame Error	Count of frame errors on the input link (only used when the data source is "External")
FrameNum	longin	@Frame Number	Reads the RTDL frame number for this channel
FrameNumSet	longout	@Frame Number	Sets the RTDL frame number for this channel
LinkError	longin	@Link Error	Count of "Input Link Lost" errors (only used when the data source is "External")

RTDL Input Module (V206) Signals for Individual Channels			
Signal Name	**Record Type**	**Function Parameter**	**Description**
Mode	bi	@Input Mode	Indicates the source of the RTDL data frame (External Link vs. VME write)
ModeSet	bo	@Input Mode	Sets the source for the RTDL data frame (External Link vs. VME write)

10.9 UTILITY MODULE SIGNALS

T.B.S.

11 REFERENCES

[1] B.Oerter, R.C.Sibley, *"System Requirements Document for Timing System"*, SNS 109020000-SR0001-R01

[2] H.Hartmann, "*Specification for the RHIC Real Time Data Link Transmitter System"*, http://www.sns.bnl.gov/epics/timing/doc/RTDL/SNS-RTDL-specs.pdf

[3] H.Hartmann, *"Specification for the V124S Beam Synchronous Trigger Module"*, http://www.sns.bnl.gov/epics/timing/doc/v124s/v124s-specs.pdf

[4] T.Kerner, *"V123S Event Link Module"*, http://www.agsrhichome.bnl.gov/Hardware/snsbeamsync/V123S.htm

[5] P.Stein, "*SNS Utility Module Functional Description"*.

[6] P.M.McGehee, *"EPICS Software for the SNS Utility Module"*.

[7] E.Bjorklund, *"Cyclic Redundancy Check Codes Used in the SNS Real-Time Data Link"*, SNS-NOTE-CNTRL-45.

[8] E.Bjorklund, *"The Theory of Rep-Rate Pattern Generation in the SNS Timing System"*, SNS-NOTE-CNTRL-99.

[9] E.Bjorklund, *"A Metric for Measuring the Evenness of Timing System Rep-Rate Patterns"*, SNS-NOTE-CNTRL-100.

[10] Joerger Enterprises Inc., *"Manual for 64 Channel, 32 Bit, Read On The Fly VME Counter/Scaler Model VS64"*, Joerger Enterprises Inc. 166 Laurel Road, East Northport, NY 11731.

[11] TrueTime Inc., *"Manuals for TrueTime XL-DC Time and Frequency Receiver and VME-SG2 GPS Interface"*, TrueTime Inc. 3750 Westwind Blvd., Santa Rosa, CA 95403.

12 CONTACTS

Brookhaven:		
Brian Oerter	oerter@bnl.gov	(631) 344-2799
Heather Hartmann	hartmann@bnl.gov	(631) 344-3071
Paul Stein	pstein@bnl.gov	(631) 344-7511
Los Alamos:		
Eric Bjorklund	bjo@lanl.gov	(505) 667-6031
Peregrine McGehee	peregrine@lanl.gov	(505) 667-3273
Oak Ridge:		
Dave Thompson	dht@ornl.gov	(865) 574-5633
Coles Sibley	sibley@ornl.gov	(865) 241-8055
Alan Jones	jonesaa@ornl.gov	(865) 241-6933

13 ABBREVIATIONS

A16, A24	VME/VXI address modes that use 16 resp. 24 address lines
AC	Alternating Current. Typically used to refer to the 60 Hz power line
CCL	Cavity Coupled Linac. The fourth accelerating section of the accelerator (after the DTL).
CRC	Cyclic Redundancy Check
DB Files	EPICS Database Files
DTL	Drift Tube Linac. The third accelerating section of the accelerator (after the RFQ).
EDM	Extensible Display Manager, an EPICS Operator Interface tool
EPICS	Experimental Physics and Industrial Control System
FIFO	First-In First-Out. A standard mechanism for queuing events
FPAR	Fast-Protect Auto Reset
FPL	Fast-Protect Latched
Frev	Ring revolution frequency. Nominally about 1.058 MHz at 1.0 GeV
GeV	Billion Electron Volts
GPS	Global Positioning System
HPRF	High Power RF
I/O	Input/Output
IOC	Input/Output Controller: Combination of front-end computer CPU, VME/VXI crate and other hardware in that crate
LANL	Los Alamos National Laboratory
LEBT	Low Energy Beam Transport
Linac	Linear Accelerator
LLRF	Low Level RF
LSB	Least Significant Byte
MeV	Million Electron Volts
MPS	Machine Protection System
MSB	Most Significant Byte
NAD	Network Attached Device. A PC-based front-end computer which controls Linac and Ring diagnostic devices.
NTP	Network Time Protocol. A protocol for synchronizing computer clocks over the internet.
OPI	Operator Interface
PLC	Programmable Logic Controller (here: Allen-Bradley Control Logix)
PLD	Programmable Logic Device
RHIC	Relativistic Heavy Ion Collider
RF	Radio Frequency
RFQ	Radio Frequency Quadripole. The second accelerating section of the accelerator (after the ion source).
SCL	Super-Conducting Linac. The fifth accelerating section of the accelerator (after the CCL).
SNL	EPICS State Notation Language
SNS	Spallation Neutron Source
TTL	Transistor-Transistor-Logic, digital signals of 0 or 5V.
VDCT	Visual Database Configuration Tool. SNS standard database building program.
VME	VERSAModule Eurocard, see http://www.vita.com. Used to refer both to the bus and compliant boards
VME64X	Extension of VME for 64 bit transfers & hot-swap

VXI .....Extension of VME Specification: requirements for standard registers etc.